TUGAS AKHIR – KI141502

PLATFORM ELEARNING UNTUK PEMBELAJARAN BAHASA PEMROGRAMAN

HAFIDH AZMI
NRP 5112100096

Dosen Pembimbing I
Dwi Sunaryono, S.Kom., M.Kom.

Dosen Pembimbing II
Rizky Januar Akbar, S.Kom., M.Eng.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR – KI141502

PLATFORM ELEARNING UNTUK PEMBELAJARAN BAHASA PEMROGRAMAN

HAFIDH AZMI
NRP 5112100096

Dosen Pembimbing I
Dwi Sunaryono, S.Kom., M.Kom.

Dosen Pembimbing II
Rizky Januar Akbar, S.Kom., M.Eng.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

ELEARNING PLATFORM FOR PROGRAMMING LANGUAGE LEARNING

HAFIDH AZMI
NRP 5112100096

Supervisor I
Dwi Sunaryono, S.Kom., M.Kom.

Supervisor II
Rizky Januar Akbar, S.Kom., M.Eng.

INFORMATICS DEPARTMENT
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PLATFORM ELEARNING UNTUK PEMBELAJARAN BAHASA PEMROGRAMAN

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma dan Pemrograman
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
HAFIDH AZMI
NRP: 5112100096

Disetujui oleh Pembimbing Tugas Akhir:

1. Dwi Sunaryono, S.Kom., M.Kom. .....
   NIP: 19720528 199702 1 001 (Pembimbing1)

2. Rizky Januar Akbar, S.Kom., M.Eng. .....
   NIP: 19870103 201404 1 001 (Pembimbing 2)

SURABAYA
JUNI, 2016

[Halaman ini sengaja dikosongkan]

PLATFORM ELEARNING UNTUK PEMBELAJARAN BAHASA PEMROGRAMAN

Nama Mahasiswa	**: HAFIDH AZMI**
NRP	**: 5112100096**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dwi Sunaryono, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Rizky Januar Akbar, S.Kom., M.Eng.**

Abstrak

Dalam tugas akhir ini, dibuat platform elearning untuk pembelajaran bahasa pemrograman. Elearning akan bisa mengatur kelas pemrograman baik kursus, peserta, soal, dan melihat jawaban dari pengguna yang terdaftar.

Dalam implementasinya, aplikasi ini berupa aplikasi web yang menggunakan kerangka kerja Laravel. Aplikasi ini dilengkapi dengan modul umpan balik secara instan yang disematkan dalam editor kode yang ada. Dengan menggunakan bantuan ANTLR JavaScript Target, kode yang dituliskan oleh pengguna akan diubah ke dalam Abstract Syntax Tree untuk dideteksi kesalahan sintaksis dan konvensi gaya penulisan kodenya.

Dengan adanya aplikasi ini, diharapkan mahasiswa mengetahui riwayat dari proses penulisan kode C++ yang dilakukannya. Modul manajemen kelas diharapkan bisa memudahkan dosen dalam mengatur kelas praktikum pemrograman.

Setelah melakukan uji coba, aplikasi ini mampu merekam history dari setiap sesi penulisan kode yang dilakukan oleh mahasiswa. Editor kode yang ada mampu menampilkan umpan balik kesalahan sintaksis dan konvensi gaya penulisan kode yang dilakukan. Pengguna juga bisa mengatur kelas, kursus, peserta, soal, dan melihat jawaban dari pengguna yang terdaftar.

Kata kunci: Abstract Syntax Tree, Laravel, Konvensi Gaya Penulisan Kode, C++

ELEARNING PLATFORM FOR PROGRAMMING LANGUAGE LEARNING

Student's Name	**: HAFIDH AZMI**
Student's ID	**: 5112100096**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Dwi Sunaryono, S.Kom., M.Kom.**
Second Advisor	**: Rizky Januar Akbar, S.Kom., M.Eng.**

Abstract

In this final project, we will build elearning platform for programming language learning. This elearning will be able to manages programming class i.e. courses, users enrollment, assignments, and answers list of the enrolled users.

In the implementatation, this web application uses Laravel framework. This application equipped with instant feedback module that embedded on its code editor. With the help of ANTLR JavaScript Target, this module will transforms user code to Abstract Syntax Tree in order to looking for the syntax and code styling convention error.

With this application, hopefully the students will know the history of their C++ code writing session. And hopefully, class management module will help the lecturers / teachers to manage their programming class.

After evaluation, this application was able to record student's code writing session. Code editor inside this application was able to show the code writing feedback instantly i.e. syntax and code styling convention error. The user also was able to manage the classes, courses, users enrollment, assignments, and answers list of the enrolled users.

Keywords: Abstract Syntax Tree, Laravel, Code Styling Convention, C++

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul ***"PLATFORM ELEARNING UNTUK PEMBELAJARAN BAHASA PEMROGRAMAN".***

Pengerjaan tugas akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan tugas akhir ini, penulis dapat belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan tugas akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya tugas akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT, karena berkat ridha-Nya lah penulis dapat menyelesaikan tugas akhir ini.
2. Bapak dan Ibu penulis yang telah memberikan dukungan moral dan material serta doa yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan tugas akhir ini.
3. Bapak Dwi Sunaryono, S.Kom., M.Kom. selaku dosen pembimbing I yang telah membantu dan membimbing penulis dalam menyelesaikan tugas akhir ini.
4. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku dosen wali penulis dan dosen pembimbing II yang telah memberikan motivasi, nasehat, bimbingan, dan bantuan yang banyak kepada penulis dalam mengerjakan tugas akhir ini.
5. Bapak Dr.Eng Darlis Herumurti, S.Kom.,M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom, M.Sc. selaku koordinator TA, dan

segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Angkatan 2012, Kakak Angkatan, dan Adik Angkatan, yang telah membantu penulis memberikan semangat untuk berjuang di dunia perkuliahan Teknik Informatika ITS.
7. Teman seperjuangan sekontrakan, Satriya, Metana, King Ardhya, Fandy, Mamin, Mas Boyke, dan Wahyu yang menjadi penyemangat penulis di tahun ke empat.
8. Teman-teman Kerja Praktik di Jogja yang selalu ceria dan memberikan motivasi pada penulis.
9. Teman-teman satu kepengurusan di HMTC dan KMI Teknik Informatika yang senantiasa sabar dan memberikan motivasi pada penulis.
10. Teman seperjuangan di Sevima, DS, Anggara, dan Akbar yang membantu dan memberikan motivasi pada penulis.
11. Pembimbing di Sevima, Pak Halim, Mas Zainul, Mas Hendri, Mas Dayat, dan rekan sekantor lainnya yang senantiasa membantu penulis dalam menghadapi permasalahan.
12. Serta semua pihak yang yang telah turut membantu penulis dalam pengerjaan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juni 2016

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

DAFTAR PERSAMAAN

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dewasa ini, penggunaan *elearning* untuk proses pembelajaran banyak diterapkan di berbagai instansi pendidikan. *Elearning* dianggap bisa memenuhi tantangan pembelajaran yang terbatasi oleh masalah ruang dan waktu. Dengan *elearning*, proses pembelajaran akan lebih efektif dan efisien. Tetapi *platform elearning* yang telah ada saat ini tidak bisa menjangkau kebutuhan para pengguna yang ingin belajar tentang pemrograman secara langsung. Para pengguna hanya bisa melakukan pengunggahan video, berkas presentasi, dan sebagainya yang tidak bisa memberikan umpan balik secara langsung mengenai kode program yang telah mereka kerjakan.

Sebelumnya, telah dibuat oleh saudara Muhammad Zuhriyan Sauqi (5110100102) dalam tugas akhir yang berjudul "Kakas Pemantauan Kinerja Programmer untuk Peningkatan Proses Personal" suatu *elearning* yang dapat mengatasi permasalahan di atas. Aplikasi yang dibuat dapat mewadahi mahasiswa yang sedang berada dalam proses pembelajaran pemrograman. Aplikasi ini juga mencatat waktu yang dibutuhkan seorang mahasiswa untuk menyelesaikan suatu soal yang diberikan. Hasil akhir yang diberikan adalah riwayat seorang mahasiswa dalam menyelesaikan soal yang diberikan. Akan tetapi, aplikasi yang telah dibuat ini memiliki kelemahan, yaitu tidak memiliki modul yang bisa menangani proses pengecekan secara instan.

Dalam tugas akhir yang penulis ajukan ini, akan dibuat *platform elearning* yang dapat membantu proses pembelajaran pemrograman yang dapat menangani proses pengecekan secara instan dengan bantuan *parser* dan *worker JavaScript*. Tugas akhir ini diharapkan mampu mengatasi ketidakmampuan aplikasi *elearning* yang telah ada.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana membuat *elearning* dengan model *platform modular* yang membantu proses pembelajaran mahasiswa dalam penulisan kode program?
2. Bagaimana membuat modul manajemen kelas pada *platform elearning*?
3. Bagaimana membuat modul umpan balik instan pada editor kode yang ada di *platform elearning*?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Teknologi yang digunakan dalam pembuatan aplikasi ini adalah PHP, Compiler C++, JavaScript, CSS, dan ANTLR JavaScript Target
2. *Platform elearning* ini dikhususkan untuk bahasa C++
3. *Platform elearning* ini hanya bisa memberikan dan menyimpan umpan balik terhadap kode program konsol
4. *Platform elearning* ini hanya bisa memantau gaya penulisan kode program (*code styling convention*), waktu pengerjaan, dan *correctness (syntax)*
5. *Platform elearning* ini digunakan untuk keperluan khusus seperti praktikum pemrograman
6. *Platform elearning* ini melakukan manajemen kelas, manajemen pengguna (dosen dan mahasiswa), dan manajemen konten (soal dan jawaban)
7. *Platform elearning* ini hanya mencatat aktivitas mahasiswa dalam proses penulisan kode program yang di-*submit*
8. Modul *instant feedback* yang ada hanya bisa meng-*cover* aturan gaya penulisan yang sudah didefinisikan terlebih dahulu.

1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Membangun *platform elearning* yang dapat memberikan mahasiswa umpan balik mengenai *code styling convention* dan *correctness* yang telah ditulis secara instan
2. Membangun *platform elearning* yang dapat memberikan dosen dan mahasiswa hasil dari aktivitas pembelajaran pemrograman dari sisi *code styling convention*, *correctness*, dan waktu pengerjaan
3. Memberikan dosen aplikasi untuk mengatur praktikum secara *online*

1.5. Manfaat

Manfaat dari hasil tugas akhir ini antara lain:

1. Mengurangi kesalahan tata tulis kode pada proses pemrograman
2. Sebagai sarana mengukur kemampuan pemrograman mahasiswa

1.6. Metodologi

Ada beberapa tahap dalam proses pengerjaan tugas akhir ini. Berikut ini adalah tahap-tahap dalam pembuatan tugas akhir.

1. Studi Literatur

Tahap ini membahas pengumpulan dan literatur yang diperlukan dalam proses perancangan dan implementasi aplikasi yang dibangun. Literatur yang digunakan adalah sebagai berikut.

a. Abstract Syntax Tree (AST)
b. Kerangka kerja Laravel
c. Template AdminLTE

d. Editor Ace
e. *Parser* dan *grammar* ANTLR
f. Worker JavaScript
g. PostgreSQL DBMS

2. Perancangan dan Desain Sistem

Pada tahap ini dilakukan perancangan sistem dengan menggunakan studi literatur dan mempelajari konsep aplikasi yang akan dibuat. Dengan berbekal teori, metode, dan informasi yang sudah terkumpul pada tahap sebelumnya diharapkan dapat membantu dalam proses perancangan sistem. Tahap ini merupakan tahap yang paling penting pada bentuk awal atau *prototype* yang akan diimplementasikan.

3. Implementasi

Pada tahap ini dilakukan implementasi rancangan sistem yang telah dibuat. Tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya sehingga menjadi sebuah aplikasi dengan simulasi yang digunakan sesuai dengan apa yang telah direncanakan.

4. Uji Coba dan Evaluasi

Pada tahap ini aplikasi yang telah selesai dibuat akan diuji. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dan ketepatan sistem dalam memberikan umpan balik secara instan pada editor kode serta proses manajemen kelas.

5. Penyusunan Laporan Tugas Akhir

Pada tahap ini disusun laporan tugas akhir sebagai dokumentasi pelaksanaan tugas akhir yang mencakup seluruh konsep, teori, implementasi, dan hasil yang telah dikerjakan.

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Tinjauan Pustaka

Bab ini membahas beberapa pustaka penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka pada aplikasi.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian berdasarkan pada kasus penggunaan dan kebutuhan fungsional sistem berdasarkan skenario yang disusun oleh penulis.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi kode sumber berupa kelas-kelas yang digunakan pada saat implementasi sistem.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan literatur yang menjadi dasar pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1. AST (Abstract Syntax Tree)

AST [1] adalah sebuah representasi pohon dari struktur sintaksis abstrak dari *source code* yang ditulis dalam bahasa pemograman tertentu. Setiap *node* yang ada menunjukkan konstruksi yang terjadi pada *source code*. Berikut ini adalah contoh dari AST.
Gambar 2.1 adalah AST dari potongan kode berikut:

```
while b ≠ 0
   if a > b
      a := a − b
   else
      b := b − a
return a
```

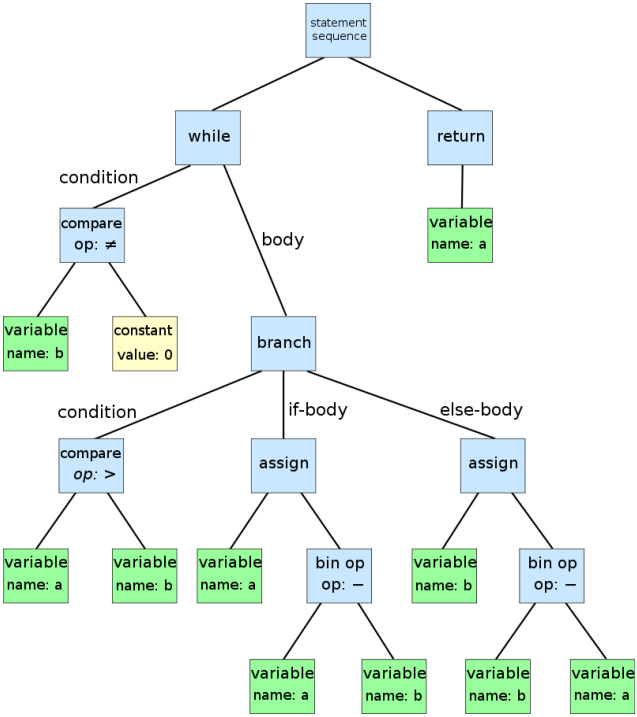
2.2. Editor Ace

Ace adalah *embedded code editor* yang ditulis dalam JavaScript. Fitur dan performanya sama dengan *native editors* seperti Sublime, Vim, dan TextMate. Ace sangat mudah disisipkan ke dalam halaman web dan aplikasi JavaScript [2].

Ace sebagai salah satu *code editor*, memiliki beberapa fitur yang seperti *code highlighter* dan *web worker*. Fitur *code hightighter* memungkinkan pewarnaan potongan kode program berdasarkan pada pembagian fungsi pada potongan kode program tersebut. Misalnya untuk deklarasi suatu *variable*, deklarasi *method, class,* akan diberikan *highlight* sehingga meningkatkan kenyamanan pada *programmer* dalam proses penulisan kode

program. Untuk tugas akhir ini, *code hightlighter* yang digunakan adalah untuk bahasa pemrograman C++.

Fitur yang kedua adalah *web worker*. *Web worker* yang disediakan oleh Editor Ace, sudah diintegrasikan dengan fitur pendeteksian penulisan kode program. Sehingga setiap *programmer* melakukan penulisan kode program, fitur ini akan memanggil *web worker* yang ada. *Web worker* pada Editor Ace inilah yang nantinya akan disambungkan dengan ANTLR JavaScript Target untuk pendeteksian *syntax* dan *code styling convention* yang ada dalam kode program yang ditulis.



Gambar 2.1 Abstract Syntax Tree

2.3. ANTLR (ANother Tool for Language Recognition)

ANTLR [3] adalah *parser generator* untuk membaca, memroses, mengeksekusi, atau menerjemahkan teks terstruktur ataupun *binary files*. ANTLR digunakan secara luas untuk membangun bahasa, kakas bantu, ataupun kerangka kerja. Dari suatu tata bahasa, ANTLR akan menghasilkan sebuah *parser* yang dapat membangun *parses trees* dan juga menghasilkan suatu *listener interface* yang dapat mempermudah pengenalan suatu frase dari suatu susunan teks. Untuk jenis ANTLR yang dipakai pada penyususan tugas akhir ini adalah ANTLR JavaScript Target yang berisi *runtime* ANTLR dalam bentuk kode program JavaScript.

2.4. Laravel

Laravel [4] adalah aplikasi *open source* yang berupa *framework* PHP dengan model MVC (*Model, View, Controller*) untuk membangun halaman web dinamis dengan menggunakan PHP. Laravel memudahkan *developer* untuk membuat aplikasi web dengan cepat dan mudah dibandingkan dengan membuatnya dari awal. *Framework* itu sendiri adalah suatu kerangka kerja yang berupa sekumpulan folder yang memuat file-file PHP yang menyediakan *class libraries*, *helpers*, *plugins*, dan lainnya. *Framework* menyediakan konfigurasi dan teknik *coding* tertentu.

2.5. AdminLTE

AdminLTE [5] adalah *template* web berbasis Bootstrap yang memiliki fitur-fitur yang cukup baik. *Template* ini terdiri dari pustaka-pustaka css dan js yang sudah didesain sedemikian rupa sehingga memiliki tampilan yang bagus dan cocok digunakan untuk halaman administrasi web. Kebutuhan akan *multi device* juga terpenuhi dengan adanya fitur *web responsive*.

2.6. Web Worker

Web worker [6] adalah suatu program JavaScript yang berjalan di belakang layar, tanpa mengganggu peforma dari halaman yang ada. *Web worker* sangat berguna apabila sistem yang dibangun melakukan proses yang cukup lama dan memiliki kemungkinan untuk mengganggu tampilan dari sistem itu sendiri. *Web worker* menggunakan konsep *request* dan *response* di mana berguna untuk mengirimkan data sebelum dan sesudah diolah.

Untuk membuat suatu kode program JavaScript dengan *web worker* [7], yang pertama kali kita lakukan adalah mendefinisikan dahulu berkas JavaScript eksternal yang akan kita panggil menggunakan *web worker*. Misalkan kita memiliki berkas JavaScript dengan nama demo_workers.js seperti pada Kode Sumber 2.1.

```
var i = 0;

function timedCount() {
   i = i + 1;
   postMessage(i);
   setTimeout("timedCount()",500);
}

timedCount();
```

Kode Sumber 2.1 Berkas JavaScript Eksternal demo_workers.js

Bagian yang paling penting dari potongan kode program di atas adalah fungsi postMessage. Fungsi ini adalah bagian dari *web worker* yang berguna untuk mengirimkan pesan dari berkas eksternal ini ke dalam berkas JavaScript yang ada pada halaman utama.

Langkah selanjutnya adalah membuat *web worker* baru pada halaman utama yang berfungsi memanggil berkas JavaScript eksternal yang disebutkan di atas, seperti yang ditulis pada Kode Sumber 2.2. Pemanggilan ini dapat dilakukan ketika terjadi suatu

event, misal terjadi pemilihan suatu tombol, penghentian penulisan, dan sebagainya.

```
w = new Worker("demo_workers.js");
```

Kode Sumber 2.2 Membuat *Web Worker* Baru

Untuk menerima pesan dari fungsi postMessage, *web worker* menggunakan fungsi yang bernama onmessage. Seperti yang tertulis pada Kode Sumber 2.3.

```
w.onmessage = function(event) {
        document.getElementById("result").innerHTML =
        event.data;
};
```

Kode Sumber 2.3 Menerima Pesan dari Berkas JavaScript Eksternal

Jika pengguna ingin mematikan *web worker* yang ada, maka pengguna harus memanggil fungsi terminate dari *web worker* tersebut. Seperti pada Kode Sumber 2.4.

```
w.terminate();
```

Kode Sumber 2.4 Mematikan *Web Worker*

2.7. Regular Expression

Regular Expression [8] adalah suatu baris dari *string* yang mendeskripsikan pola dari suatu teks tertentu. *RegEx* sangat diperlukan dalam validasi *string* yang diterima oleh sistem, agar menjaga kualitas dari masukan yang diterima. Selain itu, *RegEx* juga membantu keamanan dari sistem yaitu dengan mencegah adanya *injection* dari *script* berbahaya yang merusak sistem. *RegEx* bisa dijumpai pada semua bahasa pemrograman modern yang umum digunakan.

2.8. PostgreSQL

PostgreSQL [9] adalah suatu *database management system* yang cukup dikenal di kalangan *developer*. Selain karena *open source*, DBMS ini dianggap memiliki sistem yang kuat. DBMS ini telah dikembangkan selama 15 tahun dan telah digunakan oleh beberapa perusahaan raksasa perangkat lunak di dunia. PostgreSQL juga bisa digunakan di semua sistem operasi major yang ada. Selain itu DBMS ini juga didukung oleh bahasa pemrograman C.

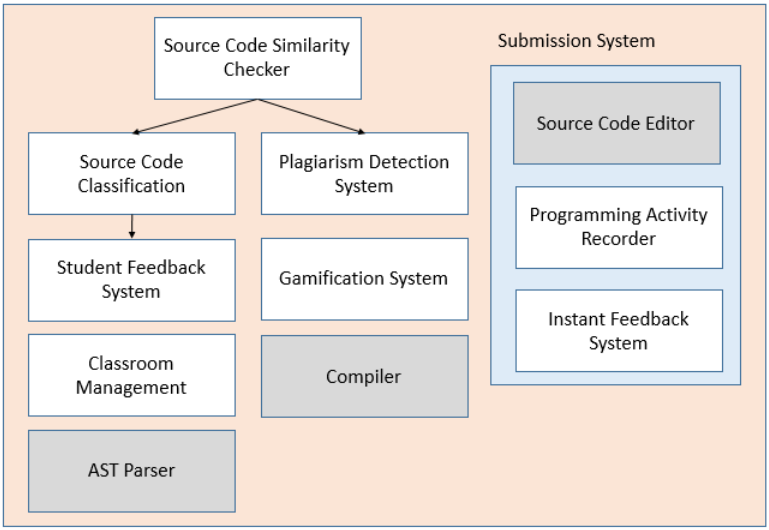
BAB III
DESAIN DAN PERANCANGAN

Pada bab ini akan dijelaskan hal-hal yang berkaitan dengan perancangan sistem yang akan dibuat dalam tugas akhir ini, dimulai dari deskripsi umum mengenai perangkat lunak yang akan dibuat, perancangan proses-proses yang ada, dan arsitektur umum sistem.

3.1. Deskripsi Umum

Dalam tugas akhir ini dibangun sebuah perangkat lunak berbasis web yang dapat mengatur pembelajaran bahasa pemrograman C++. Aplikasi ini dijalankan melalui browser pengguna. Di dalam aplikasi ini, pengguna dapat melakukan proses penulisan kode program pada editor teks yang telah dilengkapi dengan modul pendeteksian kesalahan penulisan kode dalam hal ini *syntax* dan *code styling convention* tertentu (yang telah didefinisikan sebelumnya).

Tugas akhir ini berfokus kepada *Classroom Management* dan *Instant Feedback System*. Yang dimaksudkan *Classroom Management* dalam *platform elearing* ini adalah adanya kursus yang menampung pengguna-pengguna yang telah di-*enroll* ke dalamnya. Sedangkan untuk *Instant Feedback System* dalam sistem ini adalah kolaborasi antara editor teks yang ada dengan ANTLR Javascript Parser yang dihubungkan melalui suatu *web worker*. Untuk permasalahan lain seperti pengelolaan pengguna, pengelolaan hak akses, dan pengelolaan data master, tidak akan dibahas secara rinci dalam buku tugas akhir ini.



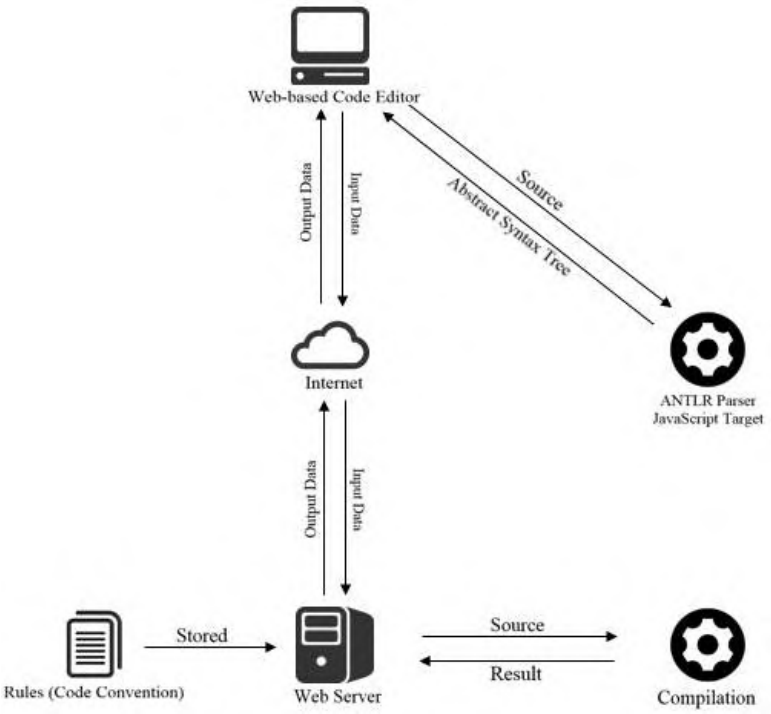
Tabel 3.1 Gambaran Sistem Secara Keseluruhan

Karena sistem ini memiliki pembagian *role* pada penggunaanya, maka kebutuhan non-fungsional pada sistem ini adalah terkait dengan keamanan (hak akses). Masing-masing pengguna akan dibagi hak aksesnya terhadap masing-masing kebutuhan fungsional yang ada.

3.2. Arsitektur Sistem

Dalam sistem ini setiap komponen memiliki peran masing-masing dalam sistem. Komponen sistem terdiri sebagai berikut:

- Pengguna
  Pengguna adalah orang yang mengakses aplikasi ini menggunakan *browser*. Ada yang berperan sebagai administrator, dosen, dan mahasiswa.
- *Web Server*
  Perangkat komputer yang berisi *database* dan *file-file* dari aplikasi ini yang sementara hanya dapat diakses melalui jaringan lokal.



Gambar 3.1 Arsitektur Sistem

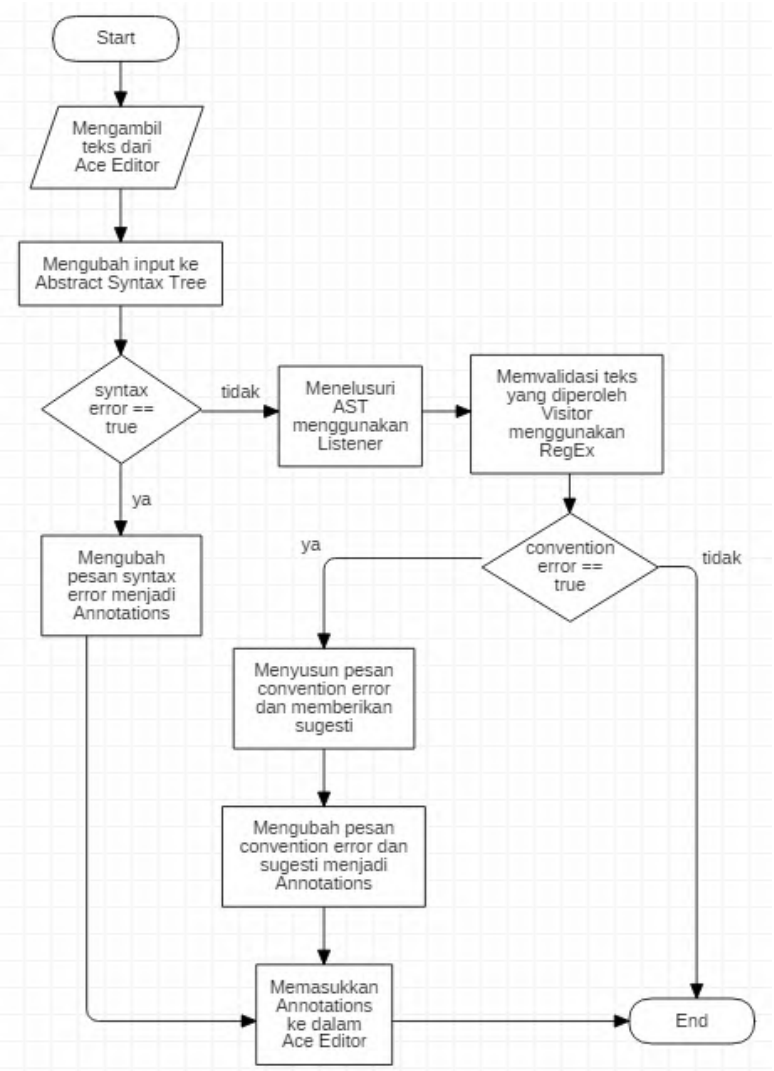
Adapun perancangan arsitektur sistem yang telah dibuat dapat dilihat pada Gambar 3.1. Adapun perancangan alur sistem yang dibangun tugas akhir ini adalah sebagai berikut:

1. Administrator membuat matakuliah dan membuat kursus sebagai wadah pengguna yang akan *enroll* ke dalam kursus tersebut.
2. Administrator mendaftarkan dosen dan mahasiswa ke dalam sistem dan memasukkan mereka ke dalam kelas kursus yang ada.
3. Dosen membuat penugasan dan detailnya, baik berupa deskripsi dan jawaban yang ada.

4. Mahasiswa melakukan proses menjawab penugasan yang ada. Bisa dilakukan beberapa kali.
5. Selama dosen dan mahasiswa menulisan kode program ke dalam editor, sistem akan secara otomatis melakukan pendeteksian kesalahan *syntax* dan *code styling convention* yang terjadi.
6. Data yang dimasukkan selama mahasiswa melakukan suatu sesi menjawab penugasan akan direkam dan akan ditampilkan riwayatnya.

3.3. Perancangan Proses *Instant Feedback*

Dalam proses pendeteksian, yang pertama kali dilakukan oleh sistem ini adalah memonitor aktivitas mengetik yang dilakukan pengguna. Editor teks melalui *web worker* yang ada akan mengirimkan kode program yang ditulis ke ANTLR JavaScript Target untuk diubah ke dalam AST. Pendeteksian kesalahan *syntax* dilakukan saat perubahan ini dengan bantuan *grammar* yang ada. Setelah bebas dari kesalahan *syntax*, sistem akan melakukan pengecakan *code styling convention*. Dengan bantuan *listener* yang ada, setiap kelas, fungsi, dan variabel lokal yang ada akan melalui proses pengecekan menggunakan *Regular Expression* yang disimpan di dalam basis data sistem. Kesalahan akan ditampilkan ke editor kode dalam bentuk *warning*. Alur kerjanya dijelaskan dalam diagram alir Gambar 3.2.



Gambar 3.2 Diagram Alir untuk Sistem *Instant Feedback*

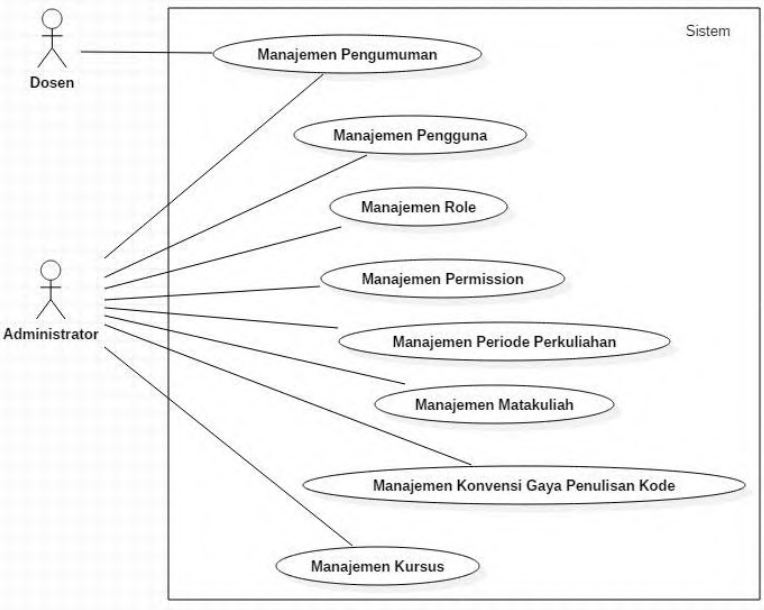
3.4. Perancangan Kasus Penggunaan

Kasus penggunaan mewakili kebutuhan fungsional sistem. Perancangan diagram kasus penggunaan ini dibagi berdasarkan pengelompokan tugas-tugas dari pengguna yang ada. Pengguna dibagi berdasarkan *role* administrator, dosen, dan mahasiswa. Administator bertugas untuk melakukan manajemen-manajemen yang diperlukan demi berjalannya aplikasi ini dengan baik. Sedangkan dosen dan mahasiswa, yang merupakan pusat dari segala tujuan diadakannya aplikasi ini, melakukan kegiatan-kegiatan yang berhubungan dengan proses pembelajaran. Oleh karena itu, berkaitan dengan proses pembelajaran yang ditujukan ke mahasiswa, kasus penggunaan dibagi menjadi kasus penggunaan non-pembelajaran dan kasus penggunaan pembelajaran.

3.4.1. Kasus Penggunaan Non-Pembelajaran

Kasus penggunaan ini hampir tidak melibatkan mahasiswa dalam alur prosesnya. Diagram kasus penggunaan dapat dilihat pada Gambar 3.3. Kebutuhan fungsional dari kasus penggunaan ini antara lain:

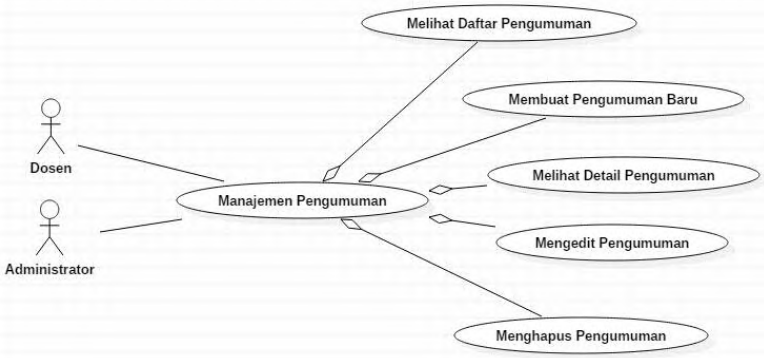
1. Manajemen Pengumuman
2. Manajemen Pengguna
3. Manajemen *Role*
4. Manajemen *Permission*
5. Manajemen Periode Perkuliahan
6. Manajemen Matakuliah
7. Manajemen Konvensi Gaya Penulisan Kode
8. Manajemen Kursus



Gambar 3.3 Kasus Penggunaan Non-Pembelajaran

3.4.1.1. Manajemen Pengumuman

Manajemen pengumuman terdiri dari beberapa kebutuhan fungsional seperti yang ditunjukkan pada Gambar 3.4.

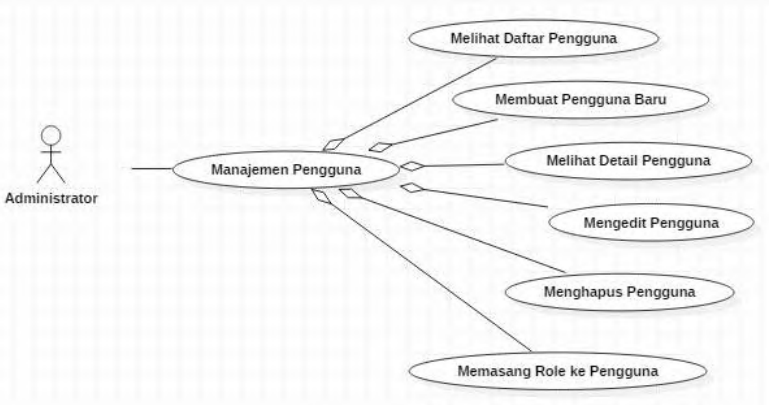


Gambar 3.4 Kasus Penggunaan Manajemen Pengumuman

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan *elearning* pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.2. Manajemen Pengguna

Manajemen pengguna terdiri dari beberapa kebutuhan fungsional seperti yang ditunjukkan pada Gambar 3.5.

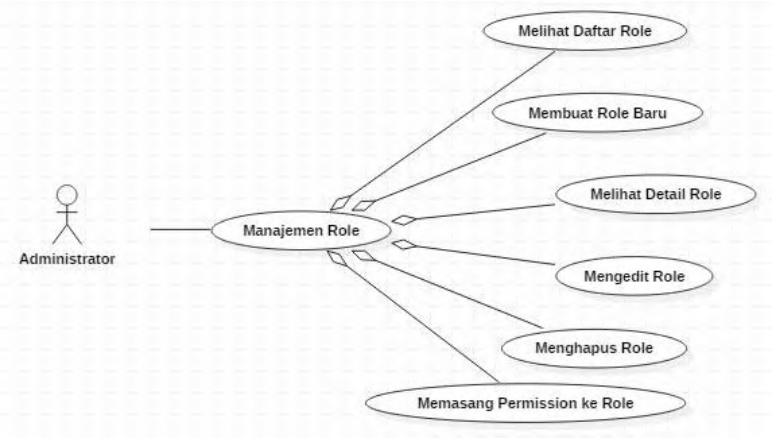


Gambar 3.5 Kasus Penggunaan Manajemen Pengguna

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan *elearning* pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.3. Manajemen *Role*

Manajemen *role* terdiri dari beberapa kebutuhan fungsional seperti yang ditunjukkan pada Gambar 3.6.

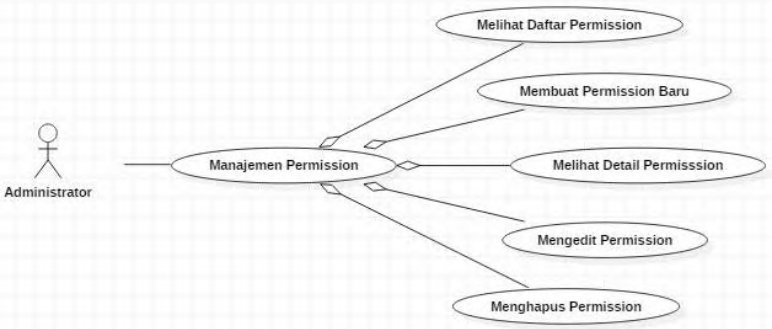


Gambar 3.6 Kasus Penggunaan Manajemen *Role*

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan *elearning* pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.4. Manajemen *Permission*

Manajemen *permission* terdiri dari beberapa kebutuhan fungsional seperti yang ditunjukkan pada Gambar 3.7.

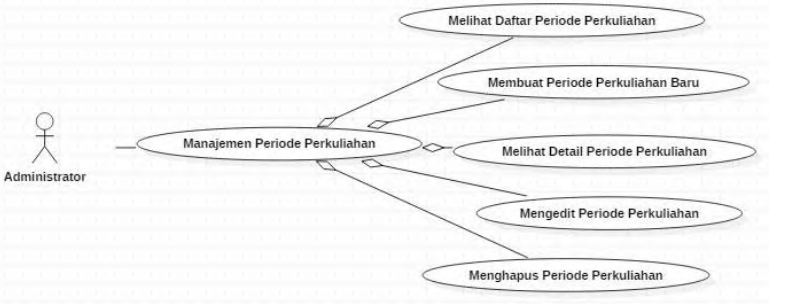


Gambar 3.7 Kasus Penggunaan Manajemen *Permission*

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan *elearning* pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.5. Manajemen Periode Perkuliahan

Manajemen periode perkuliahan terdiri dari beberapa kebutuhan fungsional seperti yang ditunjukkan pada Gambar 3.8.

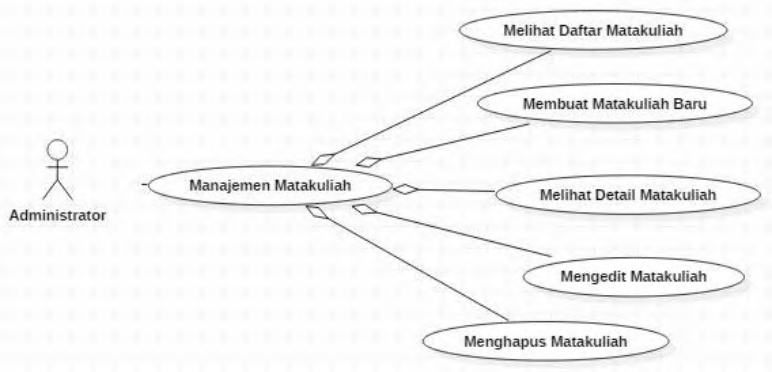


Gambar 3.8 Kasus Penggunaan Manajemen Periode Perkuliahan

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan *elearning* pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.6. Manajemen Matakuliah

Manajemen matakuliah terdiri dari beberapa kebutuhan fungsional seperti yang ditunjukkan pada Gambar 3.9.



Gambar 3.9 Kasus Penggunaan Manajemen Matakuliah

Kasus penggunaan ini bukan merupakan tujuan dari disusunnya tugas akhir ini. Tetapi sengaja dibuat sebagai bagian dari pembuatan *elearning* pada umumnya. Jadi rincian dari kasus penggunaan ini tidak akan dijelaskan dalam buku ini.

3.4.1.7. Manajemen Konvensi Gaya Penulisan Kode

Manajemen konvensi gaya penulisan kode terdiri dari beberapa kebutuhan fungsional seperti yang ditunjukkan pada Gambar 3.10.

Gambar 3.10 Kasus Penggunaan Manajemen Konvensi Gaya Penulisan Kode

Tabel berikut ini merangkum deskripsi dari masing-masing kasus penggunaan yang ada.

Tabel 3.2 Rincian Kasus Penggunaan Manajemen Konvensi Gaya Penulisan Kode

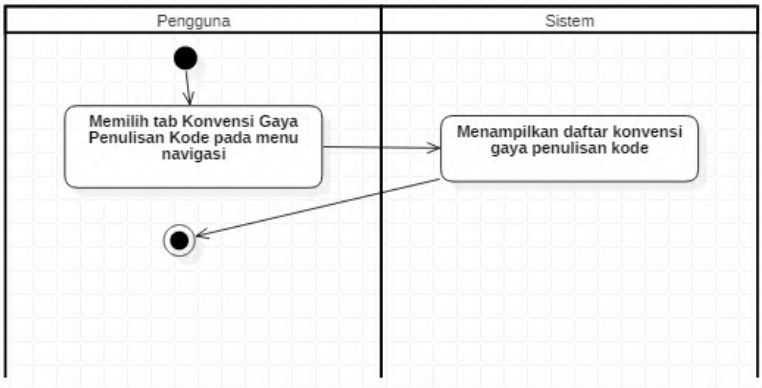
No	Kode	Nama	Keterangan
1.	UC-01	Melihat Daftar Konvensi Gaya Penulisan Kode	Aktor dapat melihat daftar konvensi gaya penulisan kode.
2.	UC-02	Membuat Konvensi Gaya Penulisan Kode Baru	Aktor dapat membuat konvensi gaya penulisan kode baru.
3.	UC-03	Melihat Detail Konvensi Gaya Penulisan Kode	Aktor dapat melihat detail dari masing-masing konvensi gaya penulisan kode.
4.	UC-04	Mengedit Konvensi Gaya Penulisan Kode	Aktor dapat mengedit konvensi gaya penulisan kode.
5.	UC-05	Menghapus Konvensi Gaya Penulisan Kode	Aktor dapat menghapus konvensi gaya penulisan kode.

3.4.1.7.1. Deskripsi Kasus Penggunaan UC-01

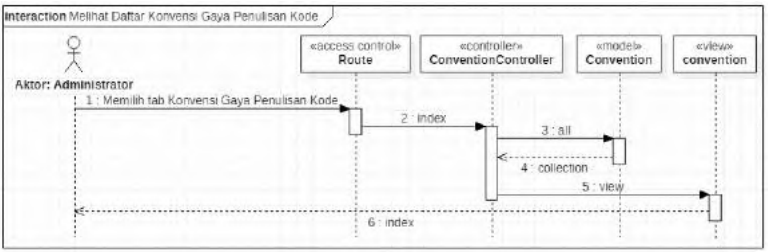
Kasus penggunaan kode UC-01 merupakan kasus penggunaan Melihat Daftar Konvensi Gaya Penulisan Kode. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.3. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.11 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.12.

Tabel 3.3 Rincian Alur Kasus Penggunaan UC-01

Nama Use Case	Melihat Daftar Konvensi Gaya Penulisan Kode
Nomor	UC-01
Aktor	Administrator
Kondisi Awal	Daftar konvensi gaya penulisan kode belum ditampilkan
Kondisi Akhir	Daftar konvensi gaya penulisan kode sudah ditampilkan
Alur Normal	1. Pengguna memilih tab Konvensi Gaya Penulisan Kode pada menu navigasi 2. Sistem menampilkan daftar konvensi gaya penulisan kode



Gambar 3.11 Diagram Aktivitas Kasus Penggunaan UC-01



Gambar 3.12 Diagram Sekuens dari Kasus Penggunaan UC-01

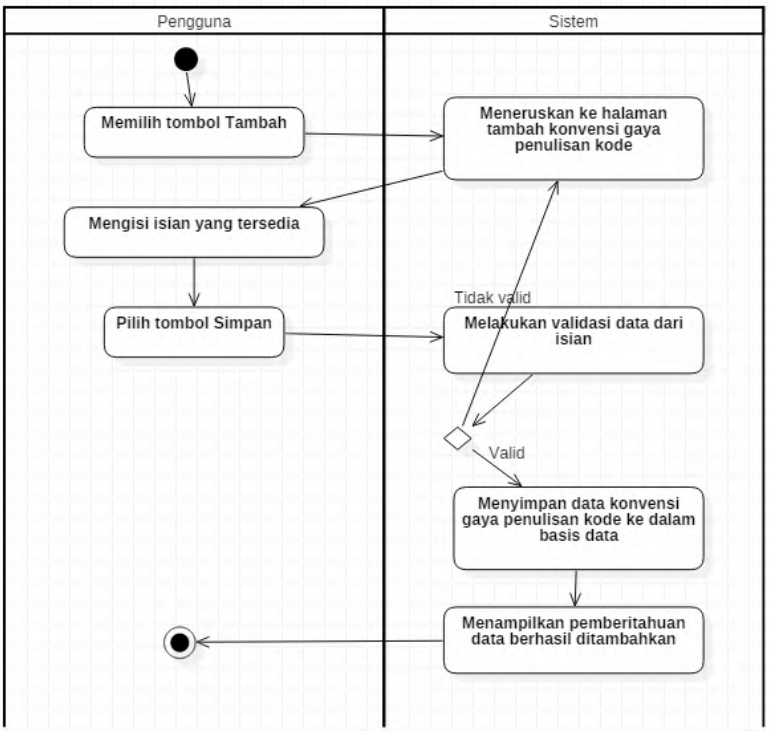
3.4.1.7.2. Deskripsi Kasus Penggunaan UC-02

Kasus penggunaan kode UC-02 merupakan kasus penggunaan Membuat Konvensi Gaya Penulisan Kode Baru. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.4. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.13 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.14.

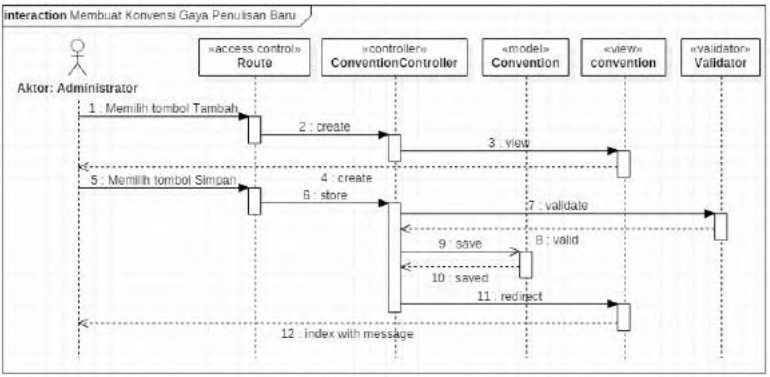
Tabel 3.4 Rincian Alur Kasus Penggunaan UC-02

Nama Use Case	Membuat Konvensi Gaya Penulisan Kode Baru
Nomor	UC-02
Aktor	Administrator
Kondisi Awal	Konvensi gaya penulisan kode baru belum ditambahkan
Kondisi Akhir	Konvensi gaya penulisan kode baru sudah ditambahkan
Alur Normal	1. Pengguna memilih tombol Tambah 2. Sistem meneruskan ke halaman tambah konvensi gaya penulisan kode 3. Pengguna mengisi isian yang tersedia 4. Pengguna memilih tombol Simpan 5. Sistem melakukan validasi data dari isian A.1. Data tidak valid

	6. Sistem menyimpan data konvensi gaya penulisan kode ke dalam basis data 7. Sistem memberikan pemberitahuan data berhasil ditambahkan
Alur Alternatif	A.1. Data tidak valid 1. Kembali ke alur normal nomor 2



Gambar 3.13 Diagram Aktivitas Kasus Penggunaan UC-02



Gambar 3.14 Diagram Sekuens dari Kasus Penggunaan UC-02

Untuk menjaga agar terjadi sinkronisasi antara *listener* yang dimiliki ANTLR dengan data yang ada di dalam basis data, maka diperlukan adanya suatu konfigurasi yang bersifat global sehingga bisa menjembatani antara basis data yang dinamis dengan *listener* bersifat *hardcode*.

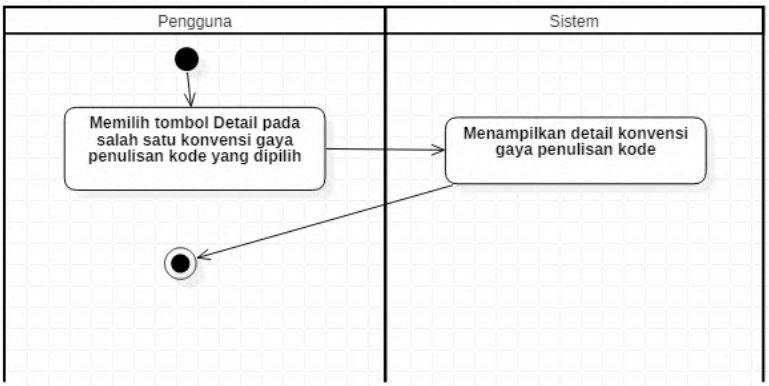
3.4.1.7.3. Deskripsi Kasus Penggunaan UC-03

Kasus penggunaan kode UC-03 merupakan kasus penggunaan Melihat Detail Konvensi Gaya Penulisan Kode. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.5. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.15 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.16.

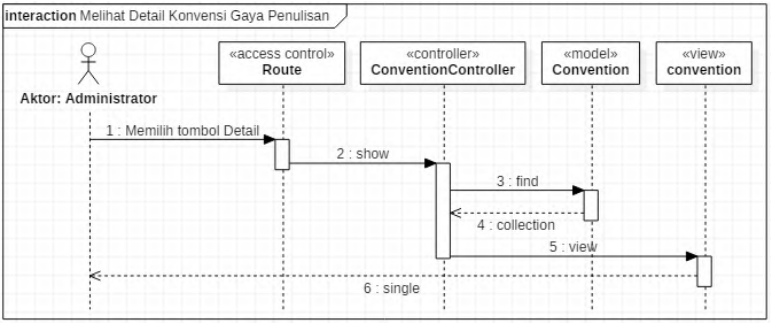
Tabel 3.5 Rincian Alur Kasus Penggunaan UC-03

Nama Use Case	Melihat Detail Konvensi Gaya Penulisan Kode
Nomor	UC-03
Aktor	Administrator
Kondisi Awal	Detail konvensi gaya penulisan kode belum ditampilkan

Kondisi Akhir	Detail konvensi gaya penulisan kode sudah ditampilkan
Alur Normal	1. Pengguna memilih tombol Detail pada salah satu konvensi gaya penulisan kode yang dipilih 2. Sistem menampilkan detail konvensi gaya penulisan kode



Gambar 3.15 Diagram Aktivitas Kasus Penggunaan UC-03



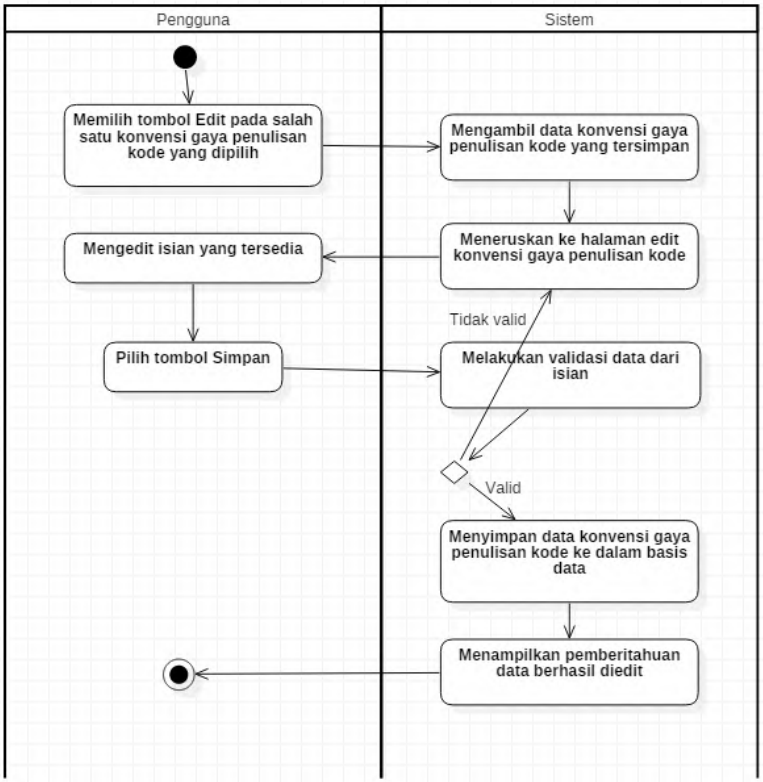
Gambar 3.16 Diagram Sekuens dari Kasus Penggunaan UC-03

3.4.1.7.4. Deskripsi Kasus Penggunaan UC-04

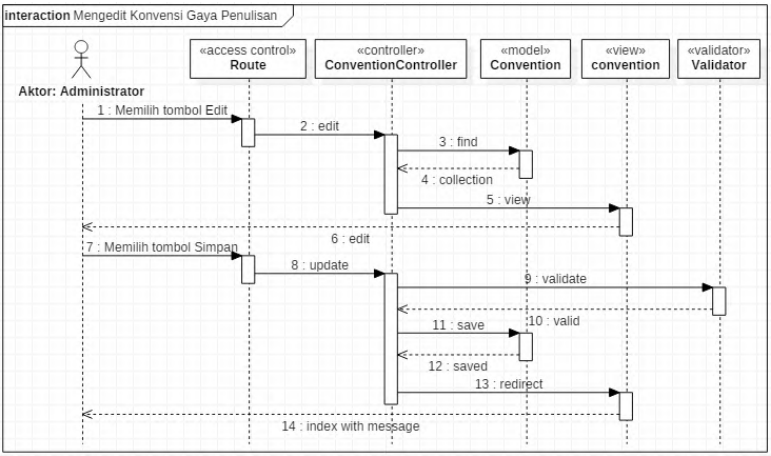
Kasus penggunaan kode UC-04 merupakan kasus penggunaan Mengedit Konvensi Gaya Penulisan Kode. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.6. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.17 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.18.

Tabel 3.6 Rincian Alur Kasus Penggunaan UC-04

Nama Use Case	Mengedit Konvensi Gaya Penulisan Kode
Nomor	UC-04
Aktor	Administrator
Kondisi Awal	Konvensi gaya penulisan kode belum diedit
Kondisi Akhir	Konvensi gaya penulisan kode sudah diedit
Alur Normal	1. Pengguna memilih tombol Edit pada salah satu konvensi gaya penulisan kode yang dipilih 2. Sistem mengambil data konvensi gaya penulisan kode yang tersimpan 3. Sistem meneruskan ke halaman edit konvensi gaya penulisan kode 4. Pengguna mengedit isian yang tersedia 5. Pengguna memilih tombol Simpan 6. Sistem melakukan validasi data dari isian A.1. Data tidak valid 7. Sistem menyimpan data konvensi gaya penulisan kode ke dalam basis data 8. Sistem menampilkan pemberitahuan data berhasil diedit
Alur Alternatif	A.1. Data tidak valid 1. Kembali ke alur normal nomor 3



Gambar 3.17 Diagram Aktivitas Kasus Penggunaan UC-04



Gambar 3.18 Diagram Sekuens dari Kasus Penggunaan UC-04

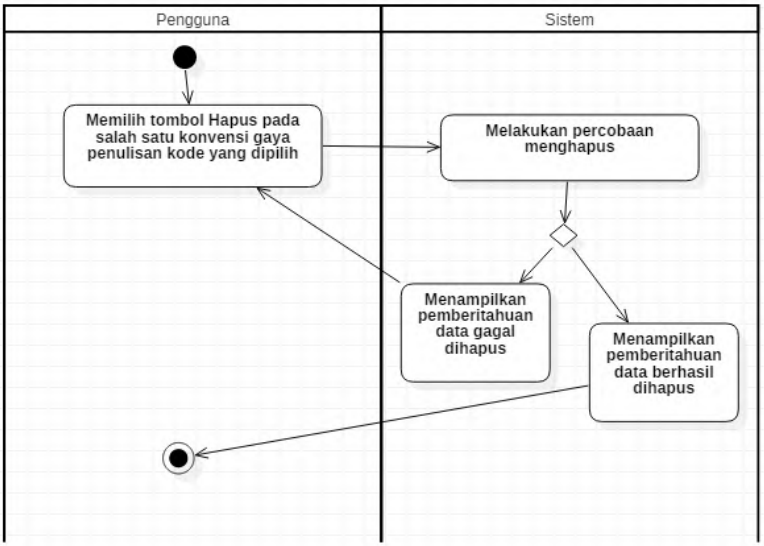
3.4.1.7.5. Deskripsi Kasus Penggunaan UC-05

Kasus penggunaan kode UC-05 merupakan kasus penggunaan Menghapus Konvensi Gaya Penulisan Kode. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.7. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.19 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.20.

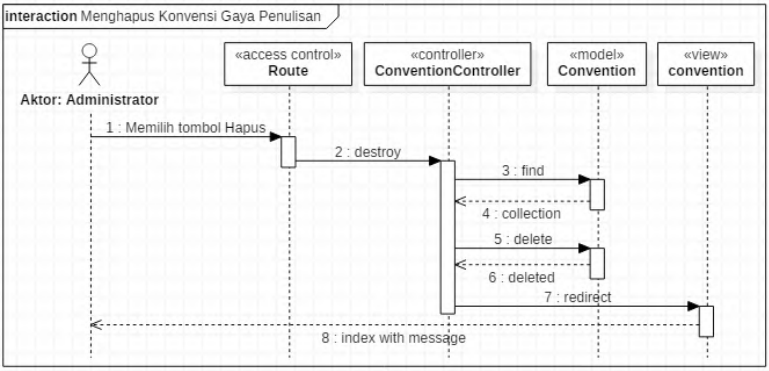
Tabel 3.7 Rincian Alur Kasus Penggunaan UC-05

Nama Use Case	Menghapus Konvensi Gaya Penulisan Kode
Nomor	UC-05
Aktor	Administrator
Kondisi Awal	Konvensi gaya penulisan kode belum dihapus
Kondisi Akhir	Konvensi gaya penulisan kode berhasil dihapus
Alur Normal	1. Pengguna memilih tombol Hapus pada salah satu konvensi gaya penulisan kode yang dipilih

	2. Sistem melakukan percobaan menghapus A.1. Percobaan menghapus gagal 3. Sistem menampilkan pemberitahuan data berhasil dihapus
Alur Alternatif	A.1. Percobaan menghapus gagal 1. Sistem menampilkan pemberitahuan data gagal dihapus 2. Kembali ke alur normal nomor 1



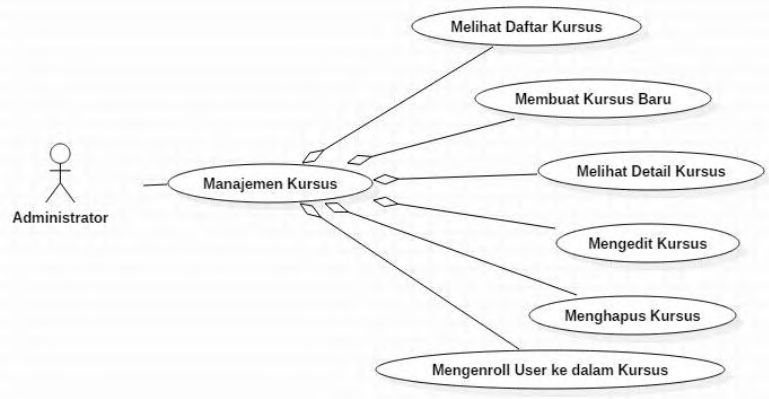
Gambar 3.19 Diagram Aktivitas Kasus Penggunaan UC-05



Gambar 3.20 Diagram Sekuens dari Kasus Penggunaan UC-05

3.4.1.8. Manajemen Kursus

Manajemen kursus terdiri dari beberapa kebutuhan fungsional seperti yang ditunjukkan pada Gambar 3.21.



Gambar 3.21 Kasus Penggunaan Manajemen Kursus

Tabel berikut ini merangkum deskripsi dari masing-masing kasus penggunaan yang ada.

Tabel 3.8 Rincian Kasus Penggunaan Manajemen Kursus

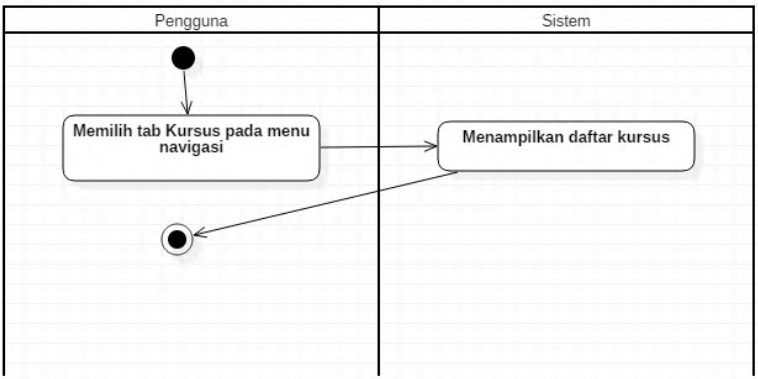
No	Kode	Nama	Keterangan
1.	UC-06	Melihat Daftar Kursus	Aktor dapat melihat daftar kursus.
2.	UC-07	Membuat Kursus Baru	Aktor dapat membuat kursus baru.
3.	UC-08	Melihat Detail Kursus	Aktor dapat melihat detail dari masing-masing kursus.
4.	UC-09	Mengedit Kursus	Aktor dapat mengedit kursus.
5.	UC-10	Menghapus Kursus	Aktor dapat menghapus kursus.
6.	UC-11	Mengenroll User ke dalam Kursus	Aktor bisa memasukkan atau mengeluarkan pengguna dari suatu kursus.

3.4.1.8.1. Deskripsi Kasus Penggunaan UC-06

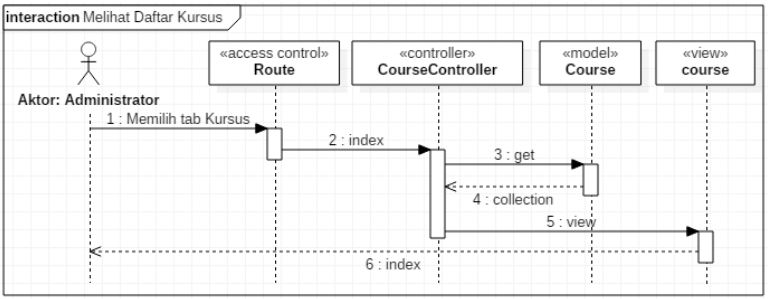
Kasus penggunaan kode UC-06 merupakan kasus penggunaan Melihat Daftar Kursus. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.9. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.22 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.23.

Tabel 3.9 Rincian Alur Kasus Penggunaan UC-06

Nama Use Case	Melihat Daftar Kursus
Nomor	UC-06
Aktor	Administrator
Kondisi Awal	Daftar kursus belum ditampilkan
Kondisi Akhir	Daftar kursus sudah ditampilkan
Alur Normal	1. Pengguna memilih tab Kursus pada menu navigasi 2. Sistem menampilkan daftar kursus



Gambar 3.22 Diagram Aktivitas Kasus Penggunaan UC-06



Gambar 3.23 Diagram Sekuens dari Kasus Penggunaan UC-06

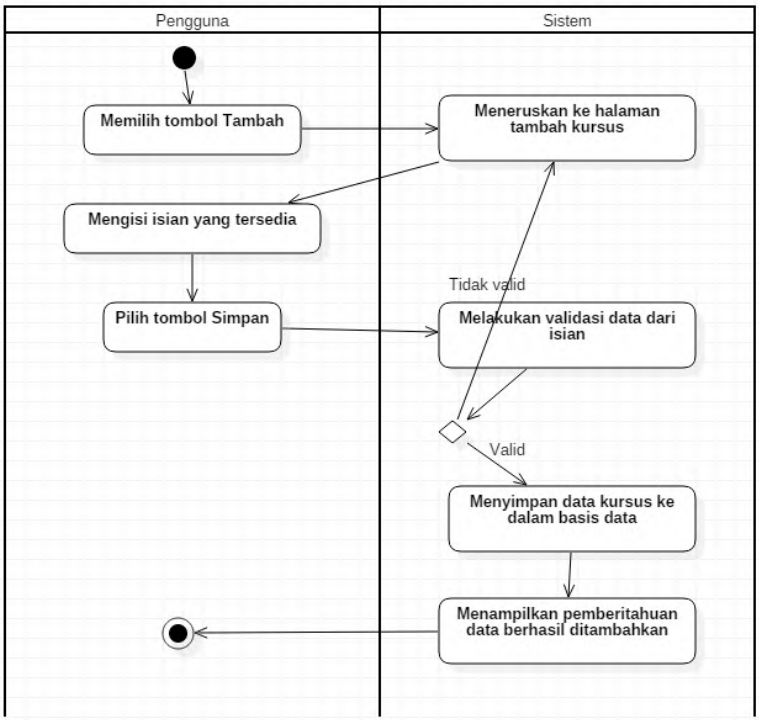
3.4.1.8.2. Deskripsi Kasus Penggunaan UC-07

Kasus penggunaan kode UC-07 merupakan kasus penggunaan Membuat Kursus Baru. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.10. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.24 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.25.

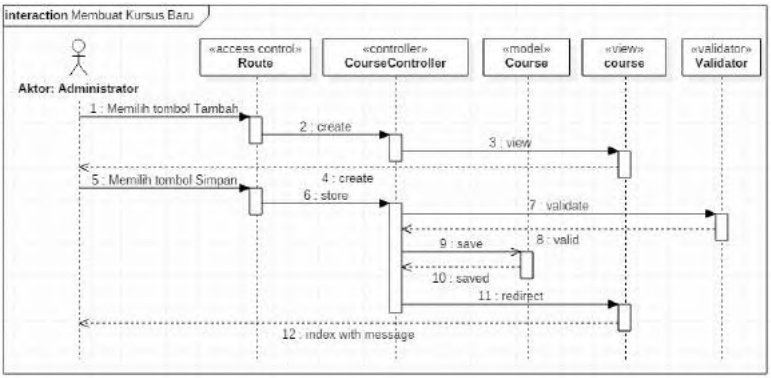
Tabel 3.10 Rincian Alur Kasus Penggunaan UC-07

Nama Use Case	Membuat Kursus Baru
Nomor	UC-07
Aktor	Administrator
Kondisi Awal	Kursus baru belum ditambahkan
Kondisi Akhir	Kursus baru sudah ditambahkan
Alur Normal	1. Pengguna memilih tombol Tambah 2. Sistem meneruskan ke halaman tambah kursus 3. Pengguna mengisi isian yang tersedia 4. Pengguna memilih tombol Simpan 5. Sistem melakukan validasi data dari isian A.1. Data tidak valid

	6. Sistem menyimpan data kursus ke dalam basis data 7. Sistem memberikan pemberitahuan data berhasil ditambahkan
Alur Alternatif	A.1. Data tidak valid 1. Kembali ke alur normal nomor 2



Gambar 3.24 Diagram Aktivitas Kasus Penggunaan UC-07



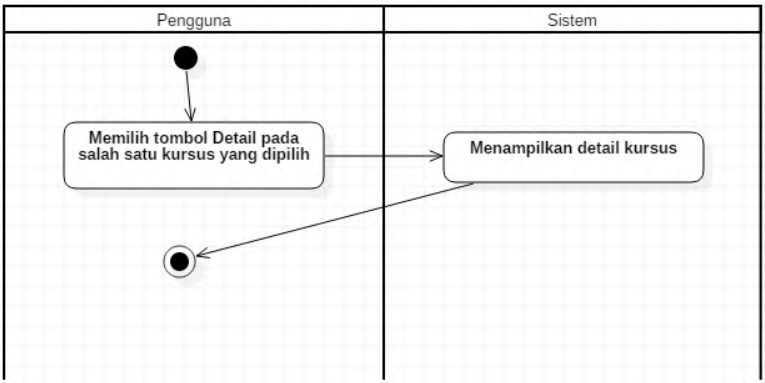
Gambar 3.25 Diagram Sekuens dari Kasus Penggunaan UC-07

3.4.1.8.3. Deskripsi Kasus Penggunaan UC-08

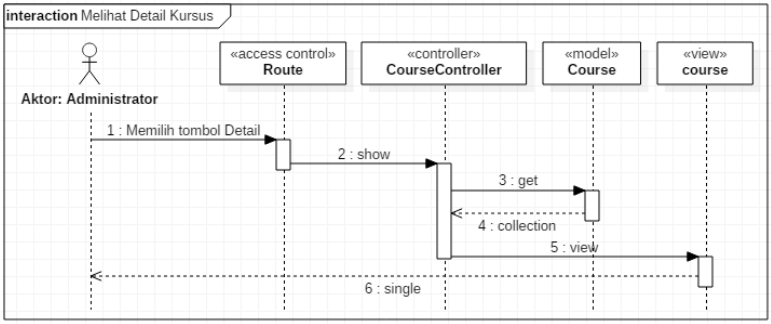
Kasus penggunaan kode UC-08 merupakan kasus penggunaan Melihat Detail Kursus. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.11. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.26 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.27.

Tabel 3.11 Rincian Alur Kasus Penggunaan UC-08

Nama Use Case	Melihat Detail Kursus
Nomor	UC-08
Aktor	Administrator
Kondisi Awal	Detail kursus belum ditampilkan
Kondisi Akhir	Detail kursus sudah ditampilkan
Alur Normal	1. Pengguna memilih tombol Detail pada salah satu kursus yang dipilih 2. Sistem menampilkan detail kursus



Gambar 3.26 Diagram Aktivitas Kasus Penggunaan UC-08



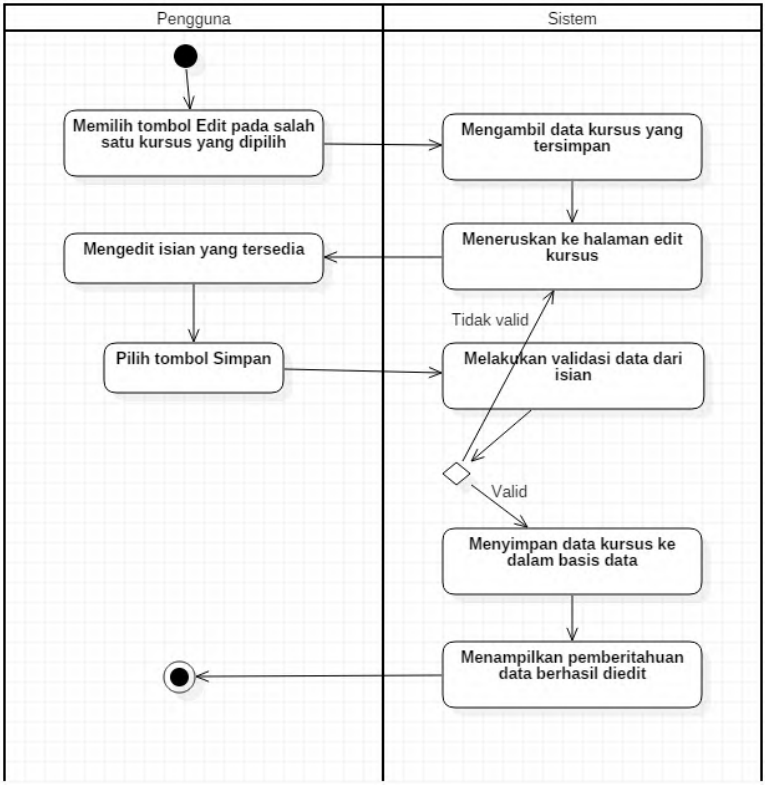
Gambar 3.27 Diagram Sekuens dari Kasus Penggunaan UC-08

3.4.1.8.4. Deskripsi Kasus Penggunaan UC-09

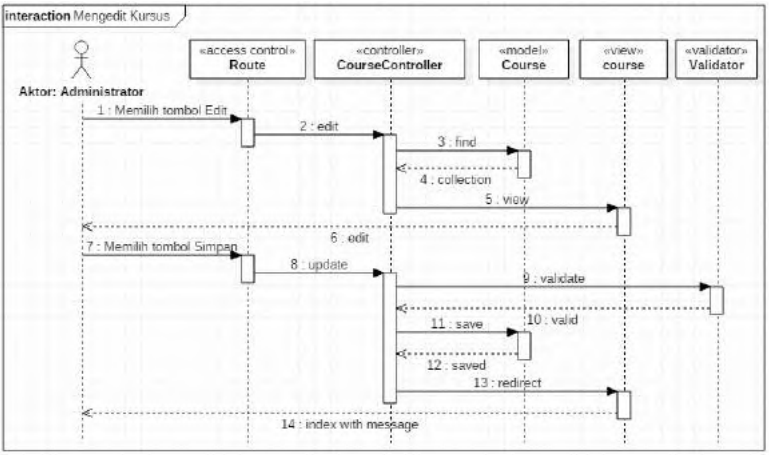
Kasus penggunaan kode UC-09 merupakan kasus penggunaan Mengedit Kursus. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.12. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.28 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.29.

Tabel 3.12 Rincian Alur Kasus Penggunaan UC-09

Nama Use Case	Mengedit Kursus
Nomor	UC-09
Aktor	Administrator
Kondisi Awal	Kursus belum diedit
Kondisi Akhir	Kursus sudah diedit
Alur Normal	1. Pengguna memilih tombol Edit pada salah satu kursus yang dipilih 2. Sistem mengambil data kursus yang tersimpan 3. Sistem meneruskan ke halaman edit kursus 4. Pengguna mengedit isian yang tersedia 5. Pengguna memilih tombol Simpan 6. Sistem melakukan validasi data dari isian A.1. Data tidak valid 7. Sistem menyimpan data kursus ke dalam basis data 8. Sistem menampilkan pemberitahuan data berhasil diedit
Alur Alternatif	A.1. Data tidak valid 1. Kembali ke alur normal nomor 3



Gambar 3.28 Diagram Aktivitas Kasus Penggunaan UC-09



Gambar 3.29 Diagram Sekuens dari Kasus Penggunaan UC-09

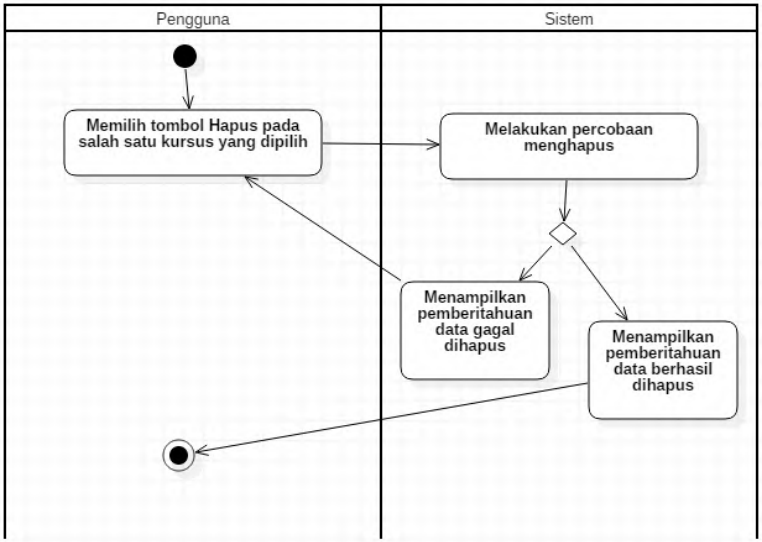
3.4.1.8.5. Deskripsi Kasus Penggunaan UC-10

Kasus penggunaan kode UC-10 merupakan kasus penggunaan Menghapus Kursus. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.13. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.30 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.31.

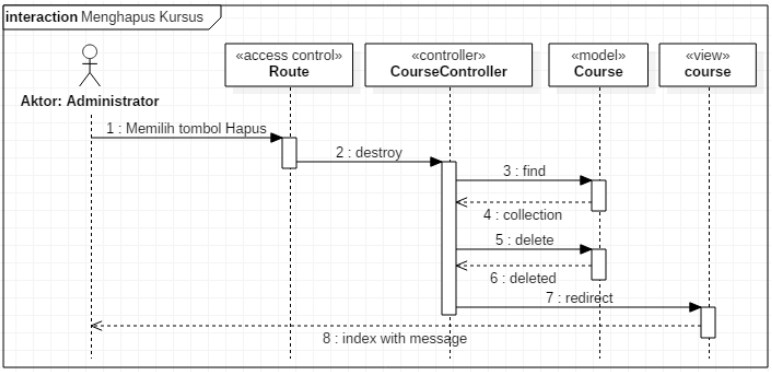
Tabel 3.13 Rincian Alur Kasus Penggunaan UC-10

Nama Use Case	Menghapus Kursus
Nomor	UC-10
Aktor	Administrator
Kondisi Awal	Kursus belum dihapus
Kondisi Akhir	Kursus berhasil dihapus
Alur Normal	1. Pengguna memilih tombol Hapus pada salah satu kursus yang dipilih 2. Sistem melakukan percobaan menghapus

	A.1. Percobaan menghapus gagal 3. Sistem menampilkan pemberitahuan data berhasil dihapus
Alur Alternatif	A.1. Percobaan menghapus gagal 1. Sistem menampilkan pemberitahuan data gagal dihapus 2. Kembali ke alur normal nomor 1



Gambar 3.30 Diagram Aktivitas Kasus Penggunaan UC-10



Gambar 3.31 Diagram Sekuens dari Kasus Penggunaan UC-10

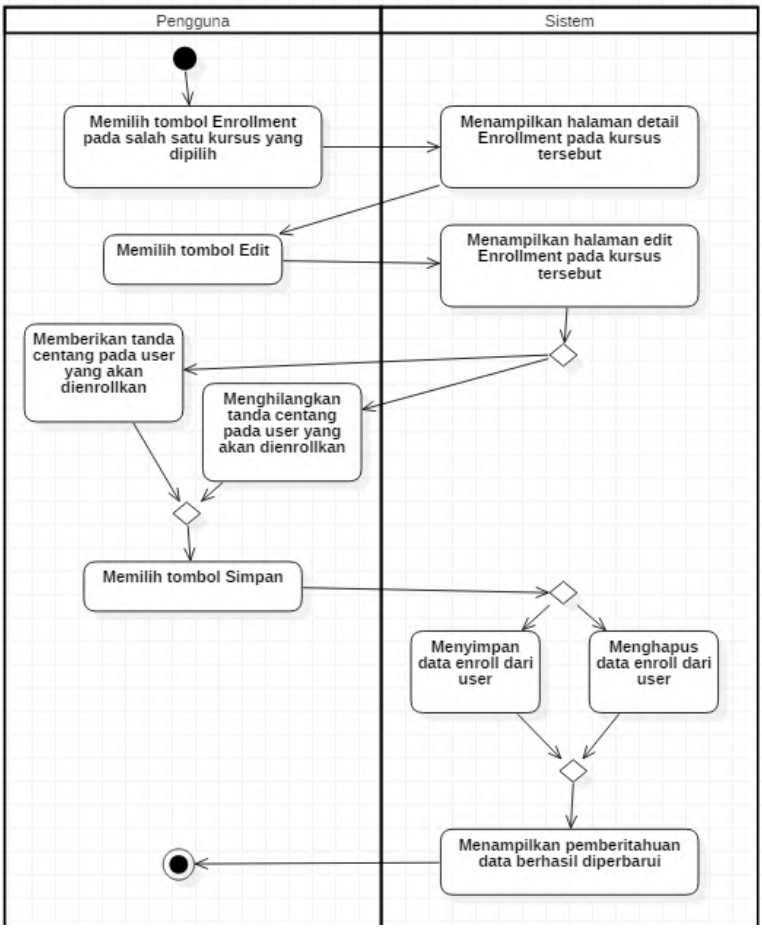
3.4.1.8.6. Deskripsi Kasus Penggunaan UC-11

Kasus penggunaan kode UC-11 merupakan kasus penggunaan Mengenroll User ke dalam Kursus. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.14. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.32 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.33.

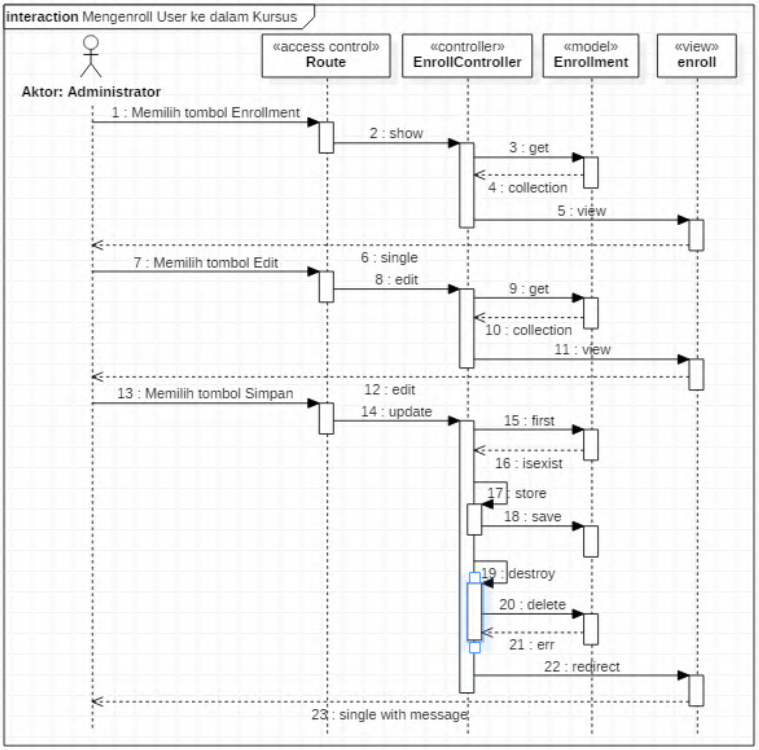
Tabel 3.14 Rincian Alur Kasus Penggunaan UC-11

Nama Use Case	Mengenroll User ke dalam Kursus
Nomor	UC-11
Aktor	Administrator
Kondisi Awal	*User* belum di-*enroll* ke dalam kursus A.1. *User* masih ada di dalam kursus
Kondisi Akhir	*User* sudah di-*enroll* ke dalam kursus A.1. *User* keluar dari kursus
Alur Normal	1. Pengguna memilih tombol Enrollment pada salah satu kursus yang dipilih 2. Sistem menampilkan halaman detail Enrollment pada kursus tersebut 3. Pengguna memilih tombol Edit

	4. Sistem menampilkan halaman edit Enrollment pada kursus tersebut 5. Pengguna memberikan tanda centang pada *user* yang akan di-*enroll*-kan A.1. Pengguna menghilangkan tanda centang pada *user* yang akan di-*unenroll*-kan 6. Pengguna memilih tombol Simpan 7. Sistem menyimpan data *enroll* dari *user* 8. Sistem menampilkan pemberitahuan data berhasil diperbarui
Alur Alternatif	A.1. Pengguna menghilangkan tanda centang pada *user* yang akan di-*unenroll*-kan 1. Pengguna memilih tombol Simpan 2. Sistem menghapus data *enroll* dari *user* 3. Kembali ke alur normal nomor 8



Gambar 3.32 Diagram Aktivitas Kasus Penggunaan UC-11



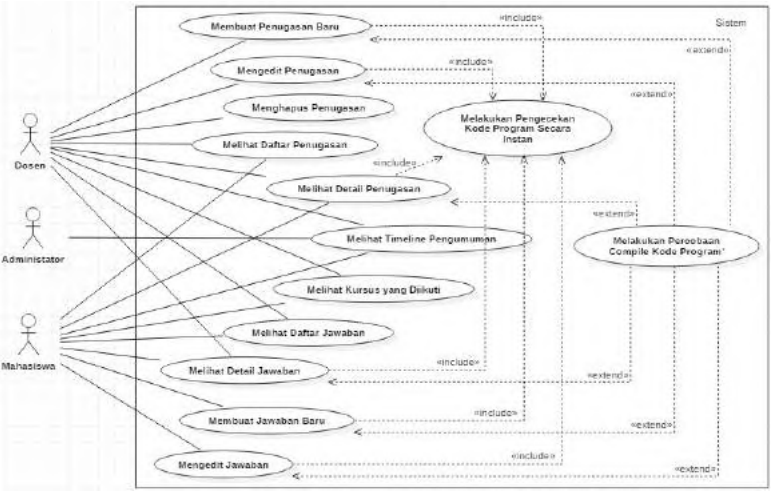
Gambar 3.33 Diagram Sekuens dari Kasus Penggunaan UC-11

3.4.2. Kasus Penggunaan Pembelajaran

Kasus penggunaan ini melibatkan mahasiswa dalam alur prosesnya. Kebutuhan fungsional dari kasus penggunaan ini antara lain:

1. Melihat Timeline Pengumuman
2. Melihat Kursus yang Diikuti
3. Melihat Daftar Penugasan
4. Membuat Penugasan Baru
5. Melihat Detail Penugasan
6. Mengedit Penugasan

7. Menghapus Penugasan
8. Melihat Daftar Jawaban
9. Membuat Jawaban Baru
10. Melihat Detail Jawaban
11. Mengedit Jawaban
12. Melakukan Pengecekan Kode Program Secara Instan
13. Melakukan Percobaan Compile Kode Program



Gambar 3.34 Kasus Penggunaan Pembelajaran

Tabel berikut ini merangkum deskripsi dari masing-masing kasus penggunaan yang ada.

Tabel 3.15 Rincian Kasus Penggunaan Pembelajaran

No	Kode	Nama	Keterangan
1.	UC-12	Melihat Timeline Pengumuman	Aktor dapat melihat pengumuman yang ada di halaman *home*
2.	UC-13	Melihat Kursus yang Diikuti	Aktor dapat melihat kursus-kursus apa saja yang mereka ikuti

3.	UC-14	Melihat Daftar Penugasan	Aktor dapat melihat daftar konvensi gaya penulisan kode.
4.	UC-15	Membuat Penugasan Baru	Aktor dapat membuat konvensi gaya penulisan kode baru.
5.	UC-16	Melihat Detail Penugasan	Aktor dapat melihat detail dari masing-masing konvensi gaya penulisan kode.
6.	UC-17	Mengedit Penugasan	Aktor dapat mengedit konvensi gaya penulisan kode.
7.	UC-18	Menghapus Penugasan	Aktor dapat menghapus konvensi gaya penulisan kode.
8.	UC-19	Melihat Daftar Jawaban	Aktor dapat melihat daftar konvensi gaya penulisan kode.
9.	UC-20	Membuat Jawaban Baru	Aktor dapat membuat konvensi gaya penulisan kode baru.
10.	UC-21	Melihat Detail Jawaban	Aktor dapat melihat detail dari masing-masing konvensi gaya penulisan kode.
11.	UC-22	Mengedit Jawaban	Aktor dapat mengedit konvensi gaya penulisan kode.
12.	UC-23	Melakukan Pengecekan Kode Program Secara Instan	Sistem dapat melakukan pengecekan kode program berdasarkan *syntax* dan *code styling convention* saat aktor melakukan proses penulisan kode program.

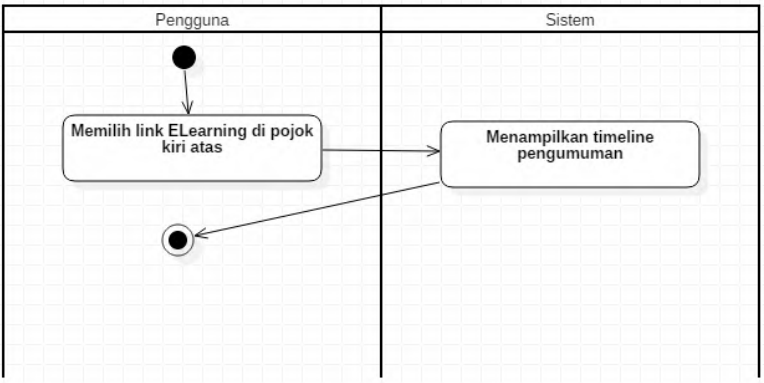
13.	UC-24	Melakukan Percobaan Compile Kode Program	Aktor dapat melakukan percobaan *compile* pada kode program yang telah ditulis.

3.4.2.1. Deskripsi Kasus Penggunaan UC-12

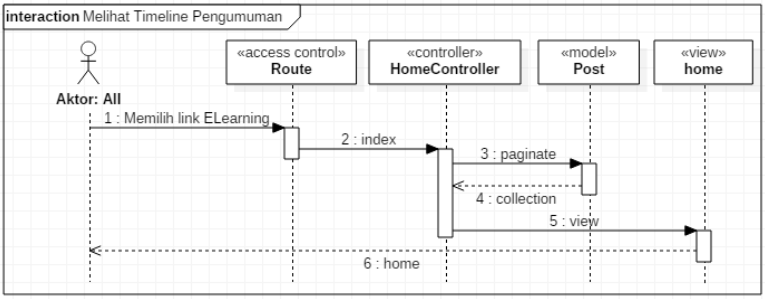
Kasus penggunaan kode UC-12 merupakan kasus penggunaan Melihat Timeline Pengumuman. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.16. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.35 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.36.

Tabel 3.16 Rincian Alur Kasus Penggunaan UC-12

Nama Use Case	Melihat Timeline Pengumuman
Nomor	UC-12
Aktor	Administrator, Dosen, Mahasiswa
Kondisi Awal	*Timeline* pengumuman belum ditampilkan
Kondisi Akhir	*Timeline* pengumuman sudah ditampilkan
Alur Normal	1. Pengguna memilih link ELearning di pojok kiri atas 2. Sistem menampilkan *timeline* pengumuman



Gambar 3.35 Diagram Aktivitas Kasus Penggunaan UC-12



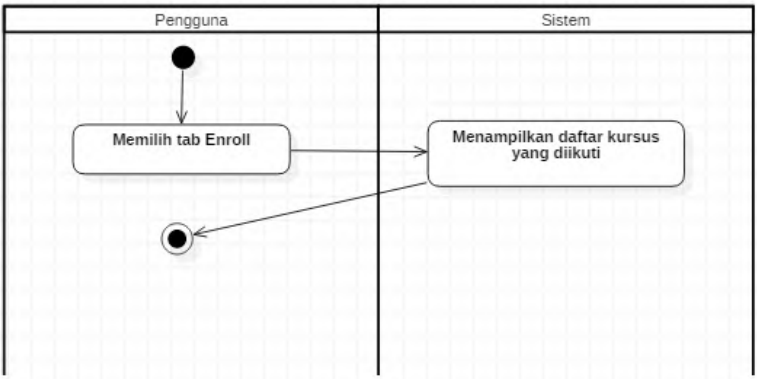
Gambar 3.36 Diagram Sekuens dari Kasus Penggunaan UC-12

3.4.2.2. Deskripsi Kasus Penggunaan UC-13

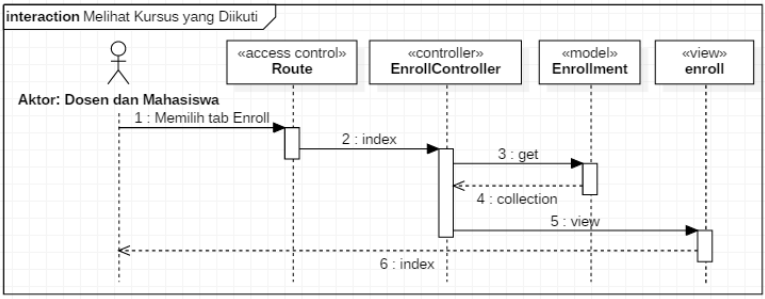
Kasus penggunaan kode UC-13 merupakan kasus penggunaan Melihat Kursus yang Diikuti. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.17. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.37 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.38.

Tabel 3.17 Rincian Alur Kasus Penggunaan UC-13

Nama Use Case	Melihat Kursus yang Diikuti
Nomor	UC-13
Aktor	Dosen, Mahasiswa
Kondisi Awal	Kursus yang diikuti belum ditampilkan
Kondisi Akhir	Kursus yang diikuti sudah ditampilkan
Alur Normal	1. Pengguna memilih tab Enroll pada menu navigasi 2. Sistem menampilkan daftar kursus yang diikuti



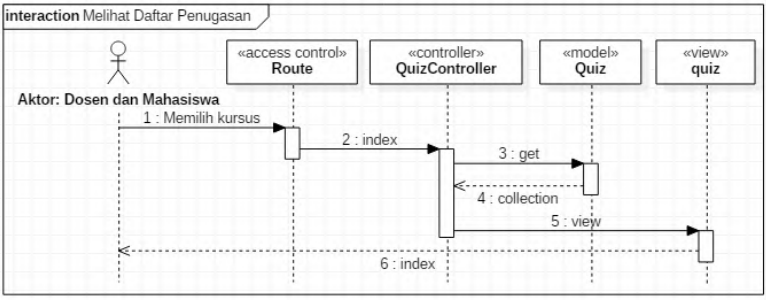
Gambar 3.37 Diagram Aktivitas Kasus Penggunaan UC-13



Gambar 3.38 Diagram Sekuens dari Kasus Penggunaan UC-13

3.4.2.3. Deskripsi Kasus Penggunaan UC-14

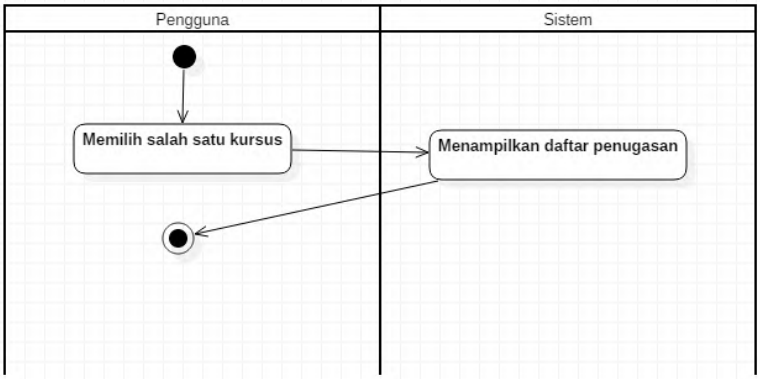
Kasus penggunaan kode UC-14 merupakan kasus penggunaan Melihat Daftar Penugasan. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.18. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.40 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.39.



Gambar 3.39 Diagram Sekuens dari Kasus Penggunaan UC-14

Tabel 3.18 Rincian Alur Kasus Penggunaan UC-14

Nama Use Case	Melihat Daftar Penugasan
Nomor	UC-14
Aktor	Dosen dan Mahasiswa
Kondisi Awal	Daftar penugasan belum ditampilkan
Kondisi Akhir	Daftar penugasan sudah ditampilkan
Alur Normal	1. Pengguna memilih salah satu kursus yang ada di halaman Enrollment 2. Sistem menampilkan daftar penugasan



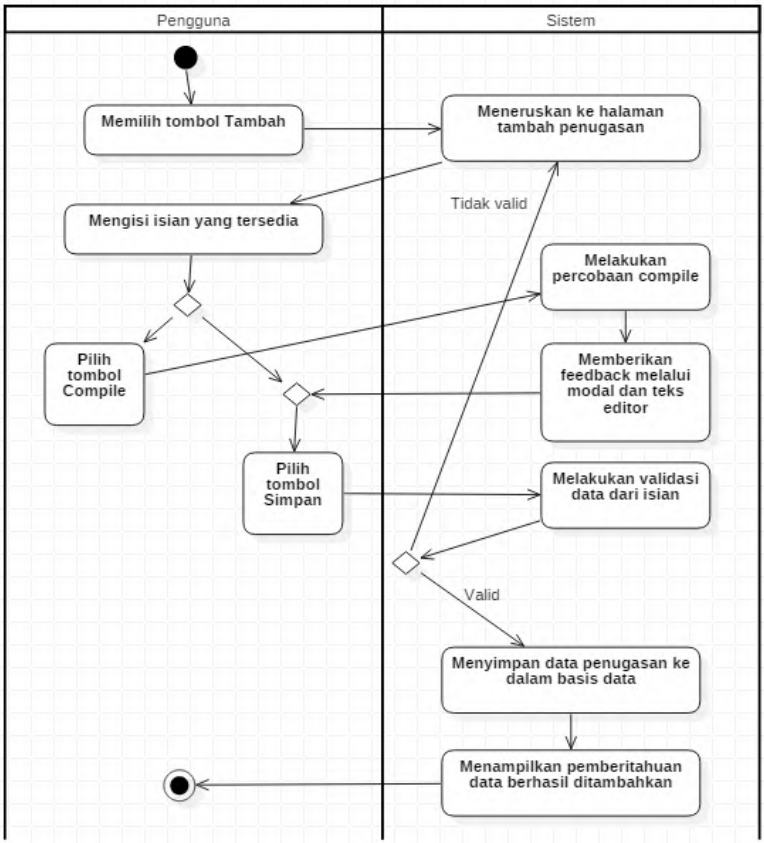
Gambar 3.40 Diagram Aktivitas Kasus Penggunaan UC-14

3.4.2.4. Deskripsi Kasus Penggunaan UC-15

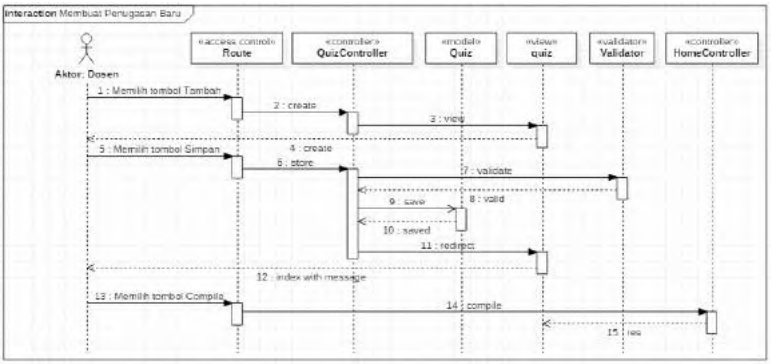
Kasus penggunaan kode UC-15 merupakan kasus penggunaan Membuat Penugasan Baru. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.19. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.41 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.42.

Tabel 3.19 Rincian Alur Kasus Penggunaan UC-15

Nama Use Case	Membuat Penugasan Baru
Nomor	UC-15
Aktor	Dosen
Kondisi Awal	Penugasan baru belum ditambahkan
Kondisi Akhir	Penugasan baru sudah ditambahkan
Alur Normal	1. Pengguna memilih tombol Tambah 2. Sistem meneruskan ke halaman tambah penugasan 3. Pengguna mengisi isian yang tersedia 4. Pengguna memilih tombol Simpan A.1. Pengguna memilih tombol Compile 5. Sistem melakukan validasi data dari isian A.2. Data tidak valid 6. Sistem menyimpan data penugasan ke dalam basis data 7. Sistem memberikan pemberitahuan data berhasil ditambahkan
Alur Alternatif	A.1. Pengguna memilih tombol Compile 1. Sistem melakukan percobaan *compile* kode program yang telah ditulis 2. Sistem memberikan *feedback* melalui modal dan editor teks 3. Kembali ke alur normal nomor 4 A.2. Data tidak valid 1. Kembali ke alur normal nomor 2



Gambar 3.41 Diagram Aktivitas Kasus Penggunaan UC-15



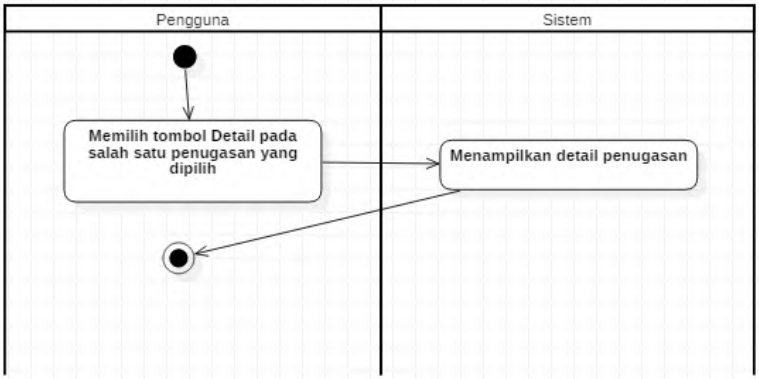
Gambar 3.42 Diagram Sekuens dari Kasus Penggunaan UC-15

3.4.2.5. Deskripsi Kasus Penggunaan UC-16

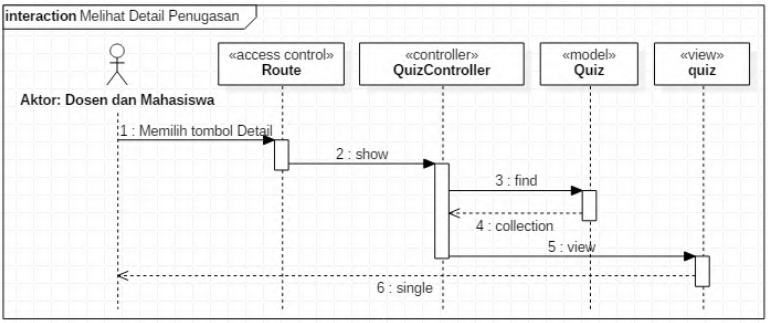
Kasus penggunaan kode UC-16 merupakan kasus penggunaan Melihat Detail Penugasan. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.20. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.43 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.44.

Tabel 3.20 Rincian Alur Kasus Penggunaan UC-16

Nama Use Case	Melihat Detail Penugasan
Nomor	UC-16
Aktor	Dosen dan Mahasiswa
Kondisi Awal	Detail penugasan belum ditampilkan
Kondisi Akhir	Detail penugasan sudah ditampilkan
Alur Normal	1. Pengguna memilih tombol Detail pada salah satu penugasan yang dipilih 2. Sistem menampilkan detail penugasan



Gambar 3.43 Diagram Aktivitas Kasus Penggunaan UC-16



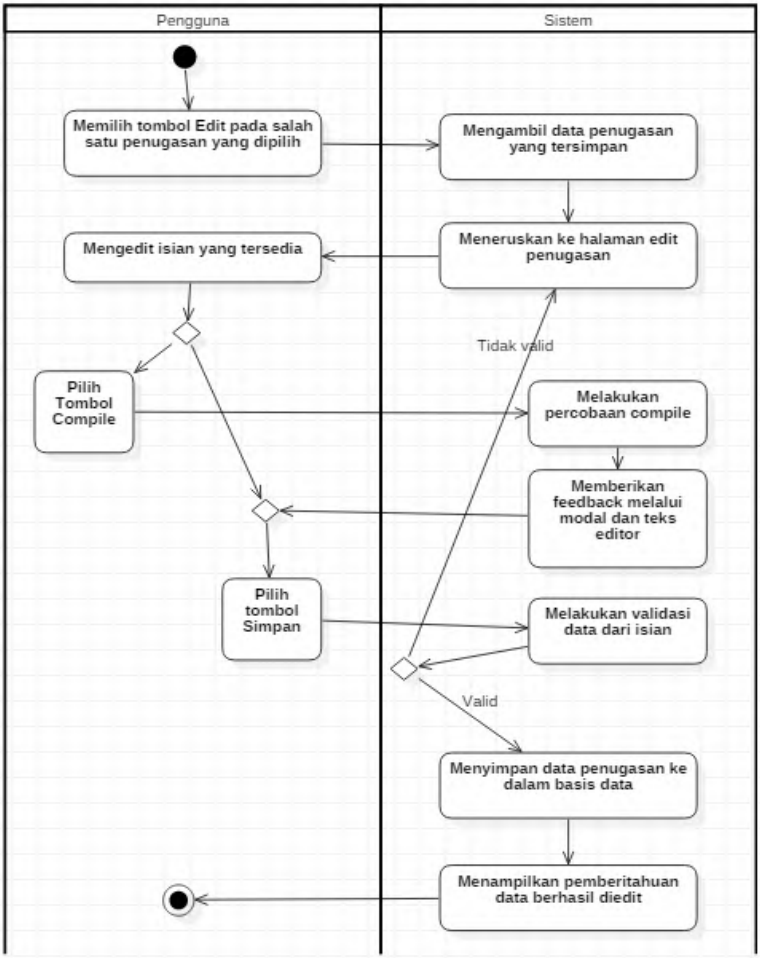
Gambar 3.44 Diagram Sekuens dari Kasus Penggunaan UC-16

3.4.2.6. Deskripsi Kasus Penggunaan UC-17

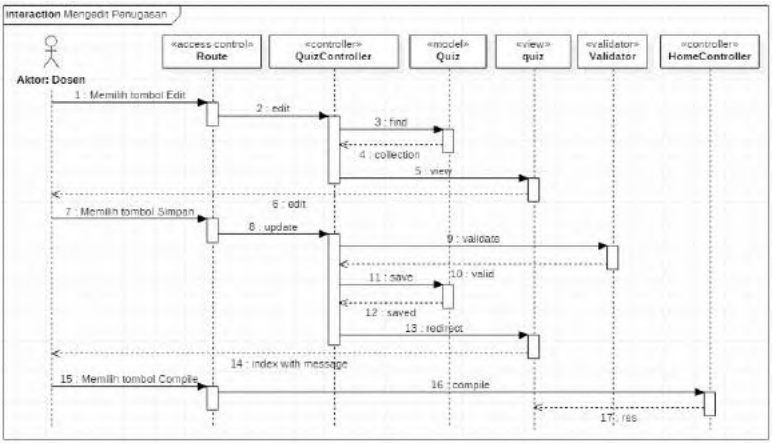
Kasus penggunaan kode UC-17 merupakan kasus penggunaan Mengedit Penugasan. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.21. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.45 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.46.

Tabel 3.21 Rincian Alur Kasus Penggunaan UC-17

Nama Use Case	Mengedit Penugasan
Nomor	UC-17
Aktor	Dosen
Kondisi Awal	Penugasan belum diedit
Kondisi Akhir	Penugasan sudah diedit
Alur Normal	1. Pengguna memilih tombol Edit pada salah satu penugasan yang dipilih 2. Sistem mengambil data penugasan yang tersimpan 3. Sistem meneruskan ke halaman edit penugasan 4. Pengguna mengedit isian yang tersedia 5. Pengguna memilih tombol Simpan A.1. Pengguna memilih tombol Compile 6. Sistem melakukan validasi data dari isian A.2. Data tidak valid 7. Sistem menyimpan data penugasan ke dalam basis data 8. Sistem menampilkan pemberitahuan data berhasil diedit
Alur Alternatif	A.1. Pengguna memilih tombol Compile 1. Sistem melakukan percobaan *compile* kode program yang telah ditulis 2. Sistem memberikan *feedback* melalui modal dan editor teks 3. Kembali ke alur normal nomor 5 A.2. Data tidak valid 1. Kembali ke alur normal nomor 3



Gambar 3.45 Diagram Aktivitas Kasus Penggunaan UC-17



Gambar 3.46 Diagram Sekuens dari Kasus Penggunaan UC-17

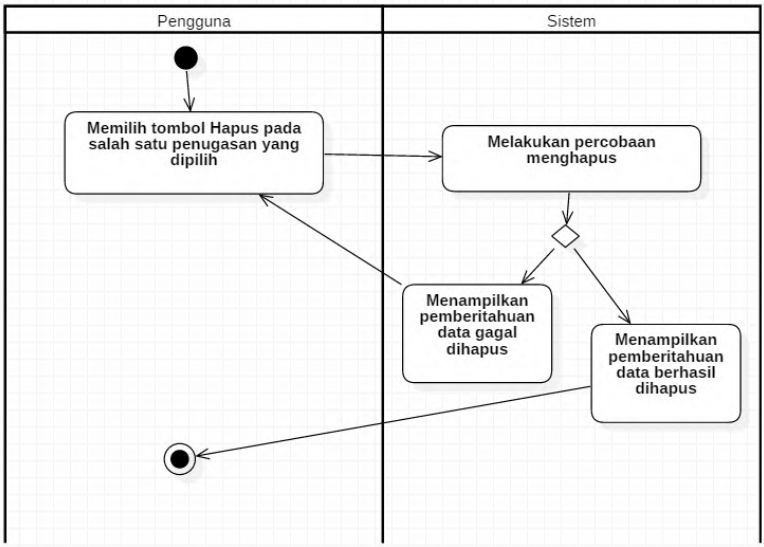
3.4.2.7. Deskripsi Kasus Penggunaan UC-18

Kasus penggunaan kode UC-18 merupakan kasus penggunaan Menghapus Penugasan. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.22. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.47 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.48.

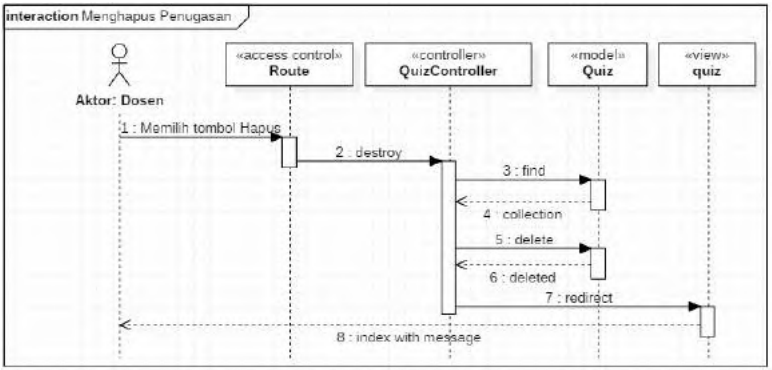
Tabel 3.22 Rincian Alur Kasus Penggunaan UC-18

Nama Use Case	Menghapus Penugasan
Nomor	UC-18
Aktor	Dosen
Kondisi Awal	Penugasan belum dihapus
Kondisi Akhir	Penugasan berhasil dihapus
Alur Normal	1. Pengguna memilih tombol Hapus pada salah satu penugasan yang dipilih 2. Sistem melakukan percobaan menghapus

	A.1. Percobaan menghapus gagal 3. Sistem menampilkan pemberitahuan data berhasil dihapus
Alur Alternatif	A.1. Percobaan menghapus gagal 1. Sistem menampilkan pemberitahuan data gagal dihapus 2. Kembali ke alur normal nomor 1



Gambar 3.47 Diagram Aktivitas Kasus Penggunaan UC-18



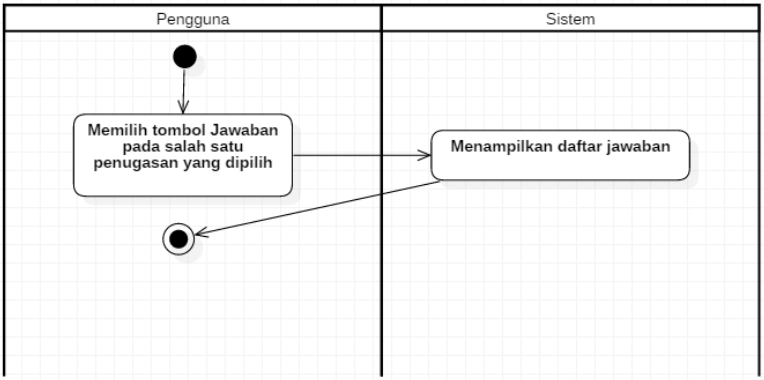
Gambar 3.48 Diagram Sekuens dari Kasus Penggunaan UC-18

3.4.2.8. Deskripsi Kasus Penggunaan UC-19

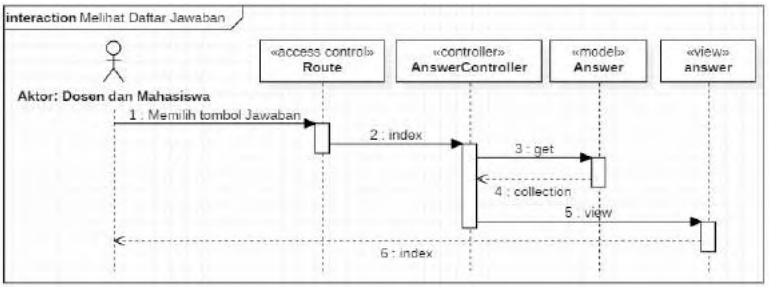
Kasus penggunaan kode UC-19 merupakan kasus penggunaan Melihat Daftar Jawaban. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.23. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.49 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.50.

Tabel 3.23 Rincian Alur Kasus Penggunaan UC-19

Nama Use Case	Melihat Daftar Jawaban
Nomor	UC-19
Aktor	Dosen dan Mahasiswa
Kondisi Awal	Daftar jawaban belum ditampilkan
Kondisi Akhir	Daftar jawaban sudah ditampilkan
Alur Normal	1. Pengguna memilih tombol Jawaban pada salah satu penugasan yang dipilih 2. Sistem menampilkan daftar jawaban



Gambar 3.49 Diagram Aktivitas Kasus Penggunaan UC-19



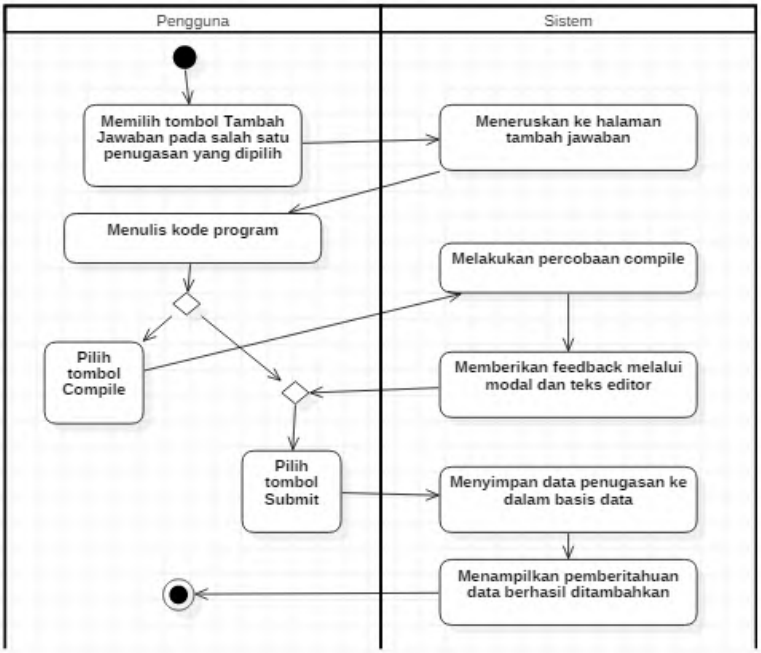
Gambar 3.50 Diagram Sekuens dari Kasus Penggunaan UC-19

3.4.2.9. Deskripsi Kasus Penggunaan UC-20

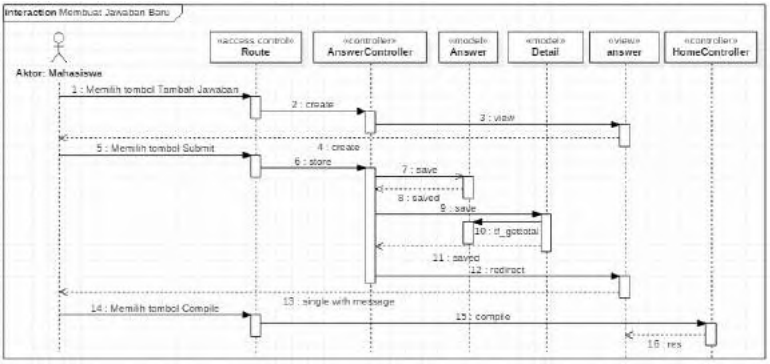
Kasus penggunaan kode UC-20 merupakan kasus penggunaan Membuat Jawaban Baru. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.24. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.51 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.52.

Tabel 3.24 Rincian Alur Kasus Penggunaan UC-20

Nama Use Case	Membuat Jawaban Baru
Nomor	UC-20
Aktor	Mahasiswa
Kondisi Awal	Jawaban baru belum ditambahkan
Kondisi Akhir	Jawaban baru sudah ditambahkan
Alur Normal	1. Pengguna memilih tombol Tambah Jawaban pada salah satu penugasan yang dipilih 2. Sistem meneruskan ke halaman tambah jawaban 3. Pengguna melakukan penulisan kode program 4. Pengguna memilih tombol Submit A.1. Pengguna memilih tombol Compile 5. Sistem menyimpan data jawaban ke dalam basis data 6. Sistem memberikan pemberitahuan data berhasil ditambahkan
Alur Alternatif	A.1. Pengguna memilih tombol Compile 1. Sistem melakukan percobaan *compile* kode program yang telah ditulis 2. Sistem memberikan *feedback* melalui modal dan editor teks 3. Kembali ke alur normal nomor 4



Gambar 3.51 Diagram Aktivitas Kasus Penggunaan UC-20



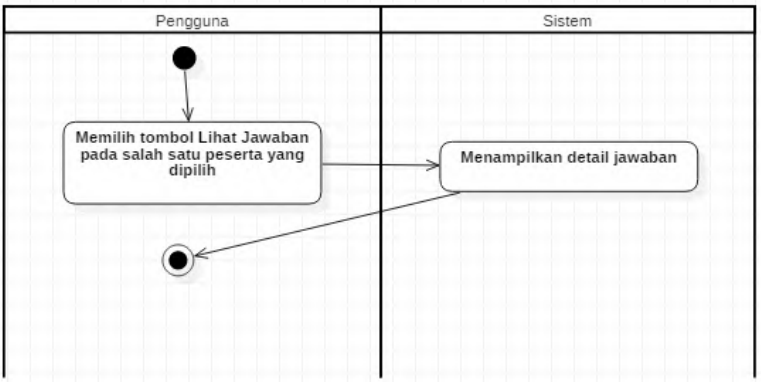
Gambar 3.52 Diagram Sekuens dari Kasus Penggunaan UC-20

3.4.2.10. Deskripsi Kasus Penggunaan UC-21

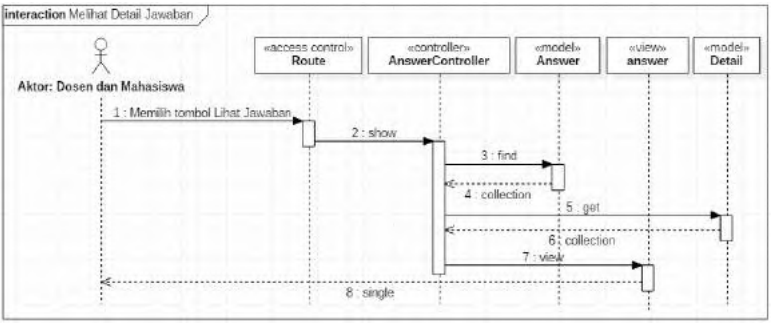
Kasus penggunaan kode UC-21 merupakan kasus penggunaan Melihat Detail Jawaban. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.25. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.53 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.54.

Tabel 3.25 Rincian Alur Kasus Penggunaan UC-21

Nama Use Case	Melihat Detail Jawaban
Nomor	UC-21
Aktor	Dosen dan Mahasiswa
Kondisi Awal	Detail jawaban belum ditampilkan
Kondisi Akhir	Detail jawaban sudah ditampilkan
Alur Normal	1. Pengguna memilih tombol Lihat Jawaban pada salah satu peserta yang dipilih 2. Sistem menampilkan detail jawaban



Gambar 3.53 Diagram Aktivitas Kasus Penggunaan UC-21



Gambar 3.54 Diagram Sekuens dari Kasus Penggunaan UC-21

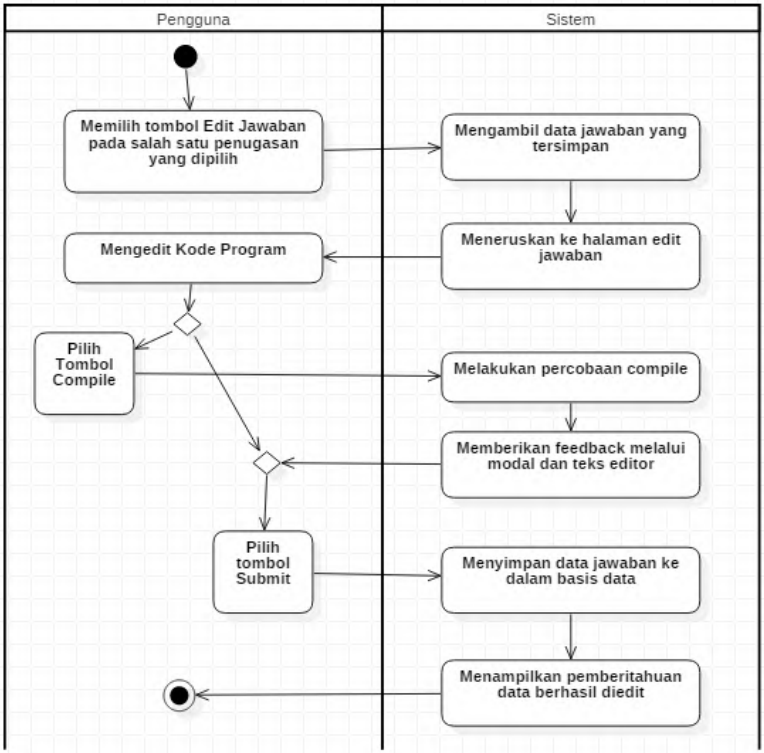
3.4.2.11. Deskripsi Kasus Penggunaan UC-22

Kasus penggunaan kode UC-22 merupakan kasus penggunaan Mengedit Jawaban. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.26. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.55 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.56.

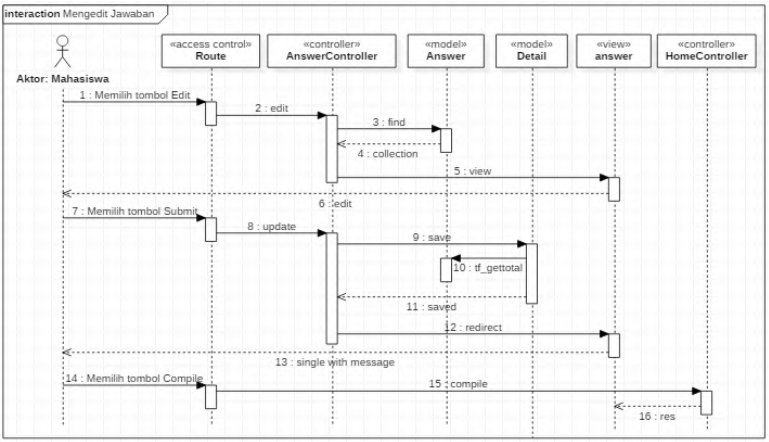
Tabel 3.26 Rincian Alur Kasus Penggunaan UC-22

Nama Use Case	Mengedit Jawaban
Nomor	UC-22
Aktor	Mahasiswa
Kondisi Awal	Jawaban belum diedit
Kondisi Akhir	Jawaban sudah diedit
Alur Normal	1. Pengguna memilih tombol Edit Jawaban pada salah satu penugasan yang dipilih 2. Sistem mengambil data jawaban yang tersimpan 3. Sistem meneruskan ke halaman edit jawaban 4. Pengguna melakukan pengeditan kode program 5. Pengguna memilih tombol Submit

	A.1. Pengguna memilih tombol Compile 6. Sistem menyimpan data jawaban ke dalam basis data 7. Sistem menampilkan pemberitahuan data berhasil diedit
Alur Alternatif	A.1. Pengguna memilih tombol Compile 1. Sistem melakukan percobaan *compile* kode program yang telah ditulis 2. Sistem memberikan *feedback* melalui modal dan editor teks 3. Kembali ke alur normal nomor 5



Gambar 3.55 Diagram Aktivitas Kasus Penggunaan UC-22



Gambar 3.56 Diagram Sekuens dari Kasus Penggunaan UC-22

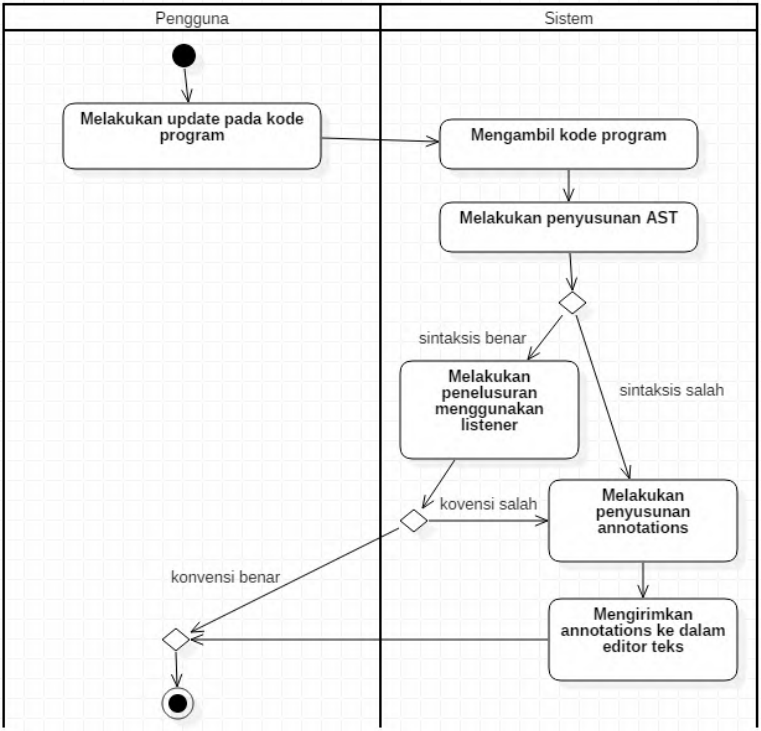
3.4.2.12. Deskripsi Kasus Penggunaan UC-23

Kasus penggunaan kode UC-23 merupakan kasus penggunaan Melakukan Pengecekan Kode Program Secara Instan. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.27. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.57 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.58.

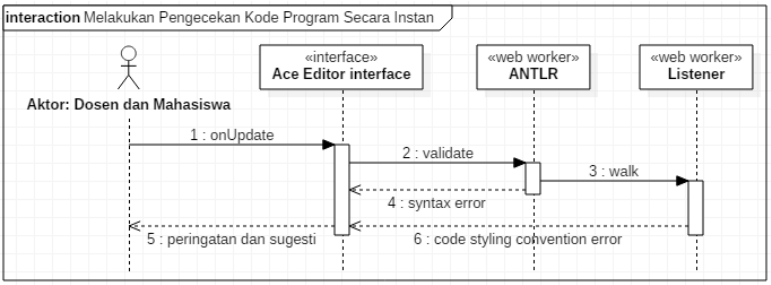
Tabel 3.27 Rincian Alur Kasus Penggunaan UC-23

Nama Use Case	Melakukan Pengecekan Kode Program Secara Instan
Nomor	UC-23
Aktor	Dosen dan Mahasiswa
Kondisi Awal	Peringatan dan koreksi belum ditampilkan
Kondisi Akhir	Tidak ada peringatan dan koreksi A.1. Peringatan dan koreksi sudah ditampilkan
Alur Normal	1. Pengguna melakukan *update* pada kode program

	2. Sistem mengambil kode program 3. Sistem melakukan penyusunan AST A.1.1. Sistem menemukan *syntax error* 4. Sistem melakukan penelusuran menggunakan *listener* A.1.2. Sistem menemukan *code styling convention error*
Alur Alternatif	A.1. Sistem menemukan *error* 1. Sistem melakukan penyusunan *annotations* 2. Sistem mengirimkan *annotations* ke dalam editor teks



Gambar 3.57 Diagram Aktivitas Kasus Penggunaan UC-23



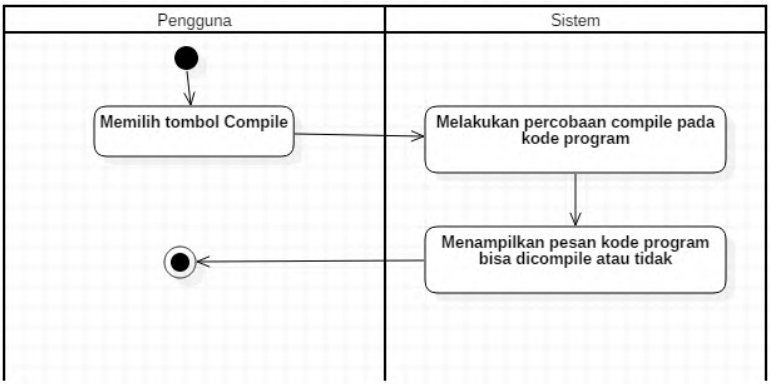
Gambar 3.58 Diagram Sekuens dari Kasus Penggunaan UC-23

3.4.2.13. Deskripsi Kasus Penggunaan UC-24

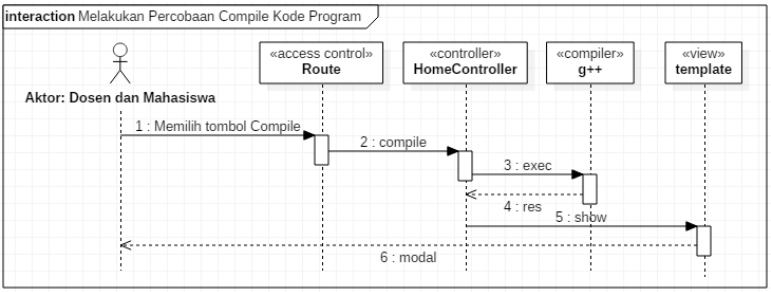
Kasus penggunaan kode UC-24 merupakan kasus penggunaan Melakukan Percobaan Compile Kode Program. Rincian alur kasus penggunaan ini dijelaskan pada Tabel 3.28. Diagram aktivitas kasus penggunaan ini dijelaskan pada Gambar 3.59 dan Diagram Sekuens dari kasus penggunaan ini dijelaskan pada Gambar 3.60.

Tabel 3.28 Rincian Alur Kasus Penggunaan UC-24

Nama Use Case	Melakukan Percobaan Compile Kode Program
Nomor	UC-24
Aktor	Dosen dan Mahasiswa
Kondisi Awal	Status *compile* belum ditampilkan
Kondisi Akhir	Status *compile* sudah ditampilkan
Alur Normal	1. Pengguna memilih tombol Compile 2. Sistem melakukan percobaan *compile* pada kode program 3. Sistem menampilkan pesan kode program bisa di-*compile* atau tidak



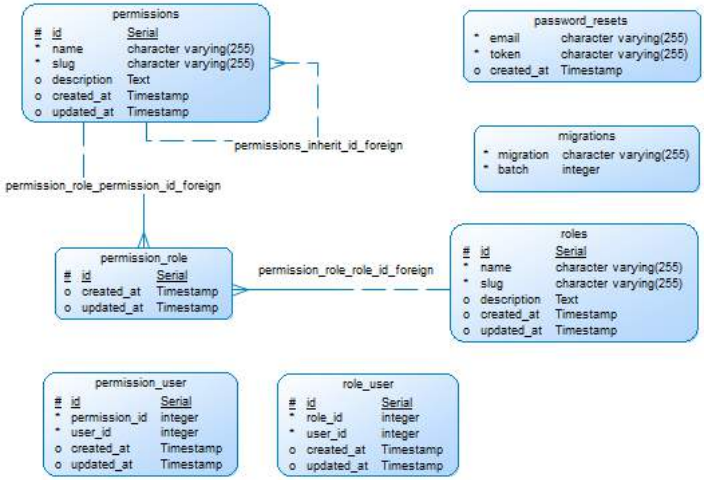
Gambar 3.59 Diagram Aktivitas Kasus Penggunaan UC-24



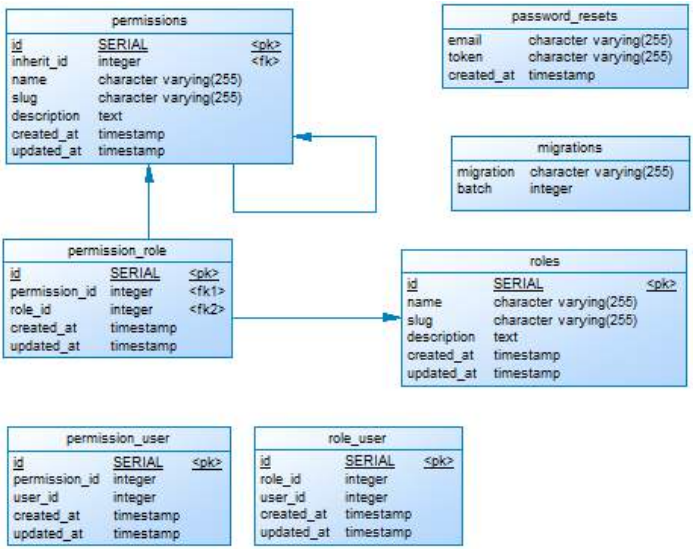
Gambar 3.60 Diagram Sekuens dari Kasus Penggunaan UC-24

3.5. Perancangan Basis Data

Perangkat lunak yang digunakan untuk mengelola dan memanajemen basis data dalam sistem ini adalah PostgreSQL. Perancangan basis data terbagi menjadi 2 yaitu basis data *access control* dan basis data *elearning.* Perancangan basis data *access control* yang berupa CDM dapat dilihat pada gambar dan PDM dapat dilihat pada gambar. Sedangkan untuk perancangan basis data *elearning* yang berupa CDM dapat dilihat pada gambar dan PDM dapat dilihat pada gambar. Untuk memperjelas basis data yang ada, dibuatlah tabel yang berisi spesifikasi basis data *access control* dan tabel yang berisikan spesifikasi basis data *elearning*.



Gambar 3.61 CDM dari Basis Data *Access Control*

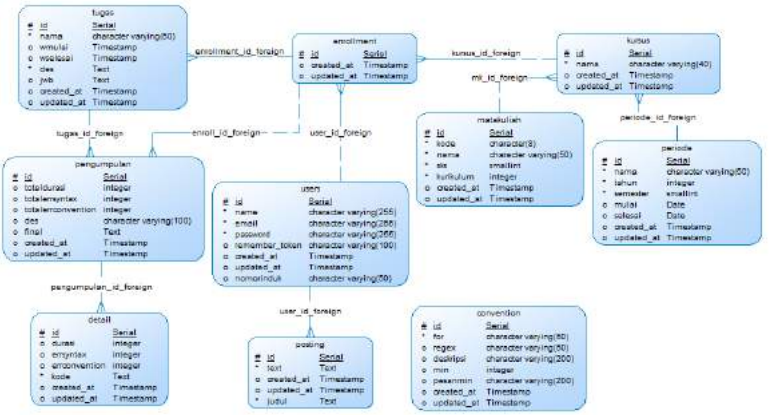


Gambar 3.62 PDM dari Basis Data *Access Control*

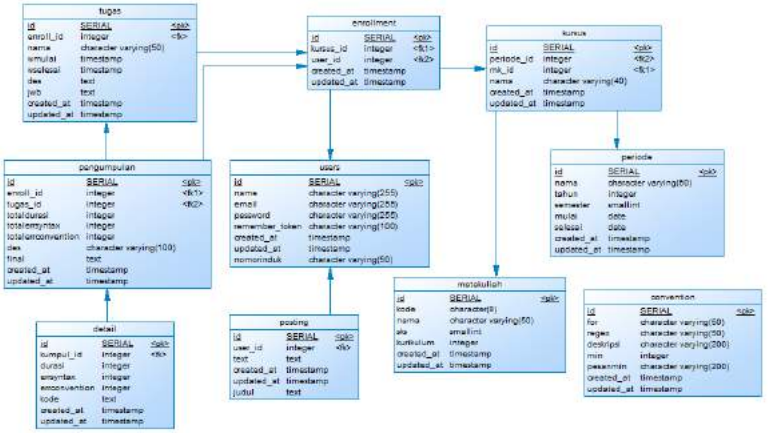
Tabel 3.29 Spesifikasi Basis Data *Access Control*

No	Tabel	Atribut	Tipe Data	Fungsi
1.	roles	id	serial	Menyimpan data *role*
		name	character varying(255)	
		slug	character varying(255)	
		description	text	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
2.	permissions	id	serial	Menyimpan data *permission*
		inherit_id	integer	
		name	character varying(255)	
		slug	character varying(255)	
		description	text	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
3.	permission_role	id	serial	Menyimpan data sinkronisasi *permission* ke dalam *role*
		permission_id	integer	
		role_id	integer	
		created_at	timestamp(0) without time zone	

		updated_at	timestamp(0) without time zone	
4.	role_user	id	serial	Menyimpan data sinkronisasi *role* ke dalam *user*
		role_id	integer	
		user_id	integer	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
5.	permission_user	id	serial	Menyimpan data sinkronisasi *permission* ke dalam *user*
		permission_id	integer	
		user_id	integer	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
6.	password_resets	email	character varying(255)	Menyimpan data password reset yang dilakukan *user*
		token	character varying(255)	
		created_at	timestamp(0) without time zone	
7.	migrations	migration	character varying(255)	Menyimpan data migrasi basis data dari laravel
		batch	integer	



Gambar 3.63 CDM dari Basis Data *Elearning*



Gambar 3.64 PDM dari Basis Data *Elearning*

Tabel 3.30 Spesifikasi Basis Data *Elearning*

No	Tabel	Atribut	Tipe Data	Fungsi
1.	users	id	serial	Menyimpan data *user*
		name	character varying(255)	
		email	character varying(255)	
		password	character varying(255)	
		remember_token	character varying(100)	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
		nomorinduk	character varying(50)	
2.	posting	id	serial	Menyimpan data pengumuman
		user_id	integer	
		text	text	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
		judul	text	
3.	periode	id	serial	Menyimpan data periode
		nama	character varying(50)	
		tahun	integer	
		semester	smallint	

		mulai	date	
		selesai	date	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
4.	matakuliah	id	serial	Meyimpan data matakuliah
		kode	character(8)	
		nama	character varying(50)	
		sks	smallint	
		kurikulum	integer	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
5.	enrollment	id	serial	Menyimpan data sinkronisasi *user* ke dalam kursus
		kursus_id	integer	
		user_id	integer	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
6.	kursus	id	serial	Menyimpan data kursus bersadarkan
		periode_id	integer	
		mk_id	integer	
		nama	character varying(40)	

		created_at	timestamp(0) without time zone	periode perkuli-ahan dan mata-kuliah
		updated_at	timestamp(0) without time zone	
7.	tugas	id	serial	Menyim-pan data penugas-an berdasar-kan pada *enroll*
		enroll_id	integer	
		nama	character varying(50)	
		wmulai	timestamp(0) without time zone	
		wselesai	timestamp(0) without time zone	
		des	text	
		jwb	text	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
8.	pengumpul an	id	serial	Menyim-pan data jawaban yang bersifat final berdasar-kan pada *enroll* dan
		enroll_id	integer	
		tugas_id	integer	
		totaldurasi	integer	
		totalerrsyntax	integer	
		totalerrconve ntion	integer	
		des	character varying(100)	
		final	text	

		created_at	timestamp(0) without time zone	penugas-an
		updated_at	timestamp(0) without time zone	
9.	detail	id	serial	Menyim-pan data jawaban yang bersifat riwayat berdasar-kan pada jawaban final yang ada
		kumpul_id	integer	
		durasi	integer	
		errsyntax	integer	
		errconvention	integer	
		kode	text	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
10.	convention	id	serial	Menyim-pan data *code styling conven-tion*
		for	character varying(50)	
		regex	character varying(50)	
		created_at	timestamp(0) without time zone	
		updated_at	timestamp(0) without time zone	
		min	integer	
		pesanmin	character varying(200)	
		deskripsi	character varying(200)	

3.6. Perancangan Antarmuka Aplikasi

Pada tugas akhir ini, antarmuka aplikasi dapat dibedakan menjadi 2 bagian utama, yaitu antarmuka manajemen dan antarmuka editor teks. Kedua antarmuka ini dapat diakses oleh pengguna menggunakan *browser* mereka.

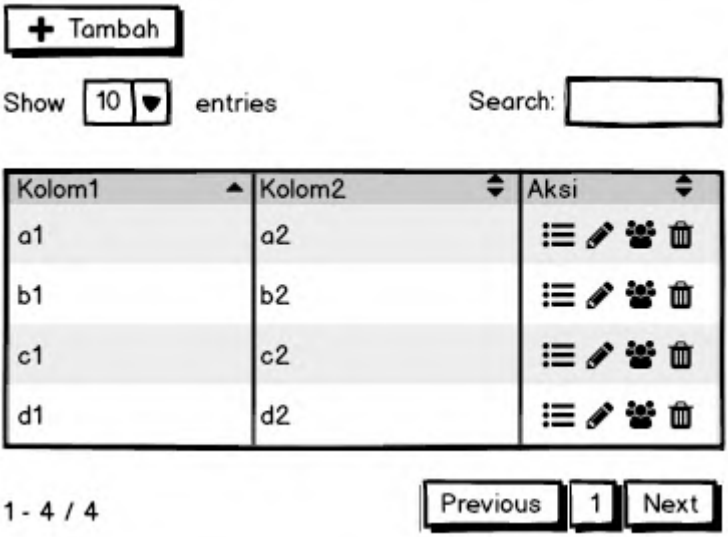
3.6.1. Antarmuka Manajemen

Antarmuka ini diimplementasikan kasus penggunaan yang melibatkan proses *create, read, update,* dan *delete* pada aplikasi. Antarmuka yang termasuk dalam CRUD antara lain adalah:

1. Antarmuka Melihat Daftar Data
2. Antarmuka Menambah Data Baru
3. Antarmuka Melihat Detail Data
4. Antarmuka Mengedit Data
5. Antarmuka Menghapus Data

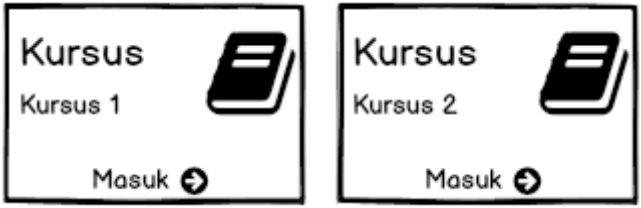
3.6.1.1. Antarmuka Melihat Daftar Data

Antarmuka ini berfungsi untuk menampilkan daftar dari data-data yang ada di dalam basis data. Antarmuka ini secara umum berbentuk *tabular* seperti pada Gambar 3.65.



Gambar 3.65 Antarmuka Melihat Daftar Data Tabular

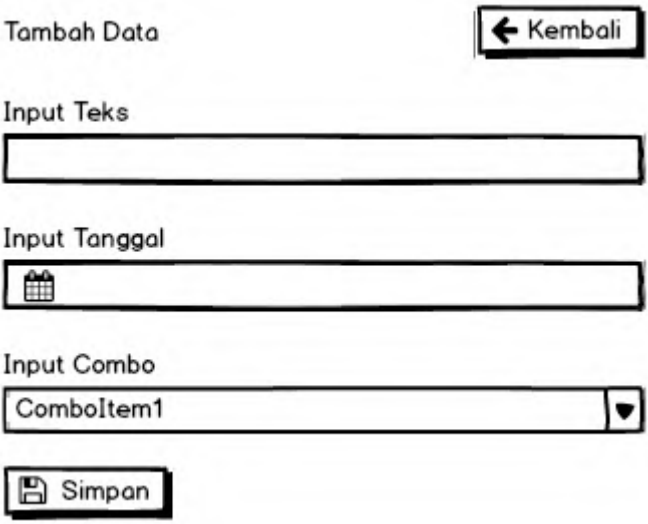
Selain bentuk *tabular*, ada juga tampilan yang berbentuk *box*, untuk setiap data yang ada. Antarmuka ini berbentuk seperti pada Gambar 3.66.



Gambar 3.66 – Antarmuka Melihat Daftar Data Box

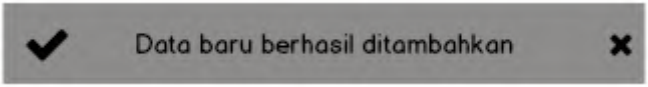
3.6.1.2. Antarmuka Menambah Data Baru

Antarmuka ini berfungsi untuk memasukkan data baru ke dalam basis data. Antarmuka ini berbentuk *form* seperti pada Gambar 3.67.



Gambar 3.67 Antarmuka Menambah Data Baru

Setelah data baru berhasil ditambahkan, akan dimunculkan pesan seperti pada Gambar 3.68.



Gambar 3.68 Antarmuka Penambahan Data Berhasil

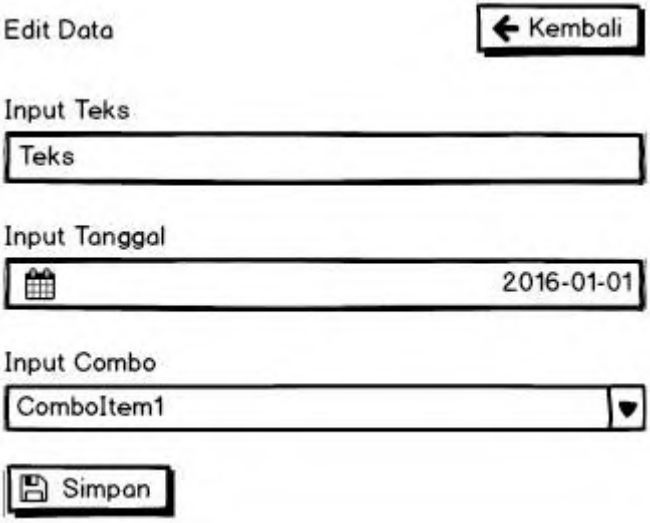
3.6.1.3. Antarmuka Melihat Detail Data

Antarmuka ini berfungsi untuk menampilkan 1 item data secara detail. Antarmuka ini berbentuk seperti pada Gambar 3.69.

Gambar 3.69 Antarmuka Melihat Detail Data

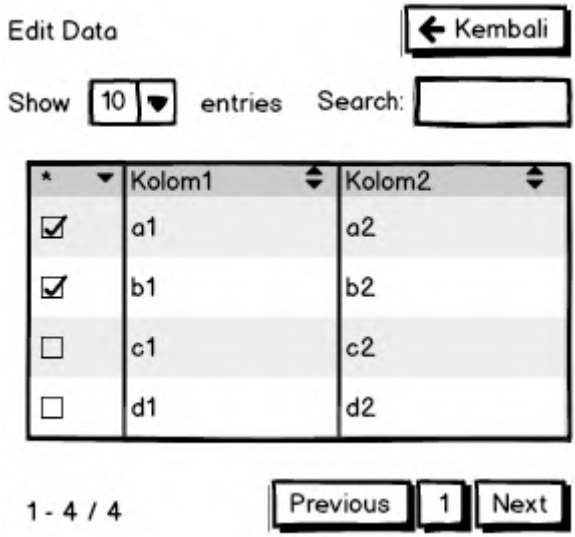
3.6.1.4. Antarmuka Mengedit Data

Antarmuka ini berfungsi untuk mengedit data yang ada di dalam basis data. Antarmuka ini berbentuk *form* seperti pada Gambar 3.70.



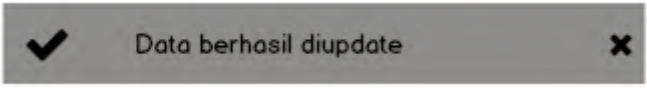
Gambar 3.70 Antarmuka Mengedit Data Form

Selain berbentuk *form* seperti di atas, ada juga yang berbentuk seperti *checklist*. Seperti pada Gambar 3.71.



Gambar 3.71 Antarmuka Mengedit Data Checklist

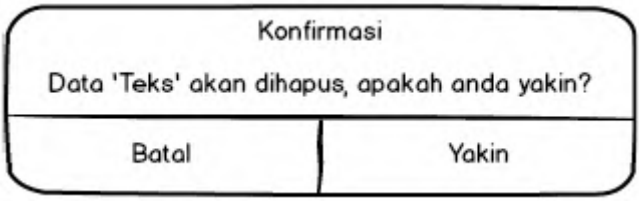
Setelah data baru berhasil diedit, akan dimunculkan pesan seperti pada Gambar 3.72.



Gambar 3.72 Antarmuka Pengeditan Data Berhasil

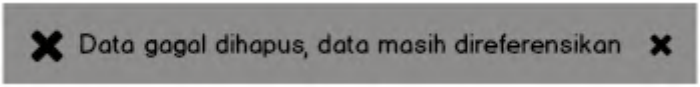
3.6.1.5. Antarmuka Menghapus Data

Antarmuka ini berfungsi untuk menghapus data yang ada di dalam basis data. Antarmuka ini berbentuk *modal* seperti pada Gambar 3.73.



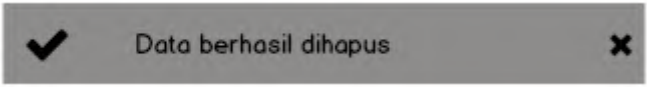
Gambar 3.73 Antarmuka Menghapus Data

Setelah konfirmasi penghapusan disetujui pengguna, sistem akan mencoba melakukan penghapusan. Jika data masih direferensikan oleh data lainnya, maka penghapusan akan gagal dan sistem menampilkan pesan seperti pada Gambar 3.74.



Gambar 3.74 Antarmuka Percobaan Penghapusan Data Gagal

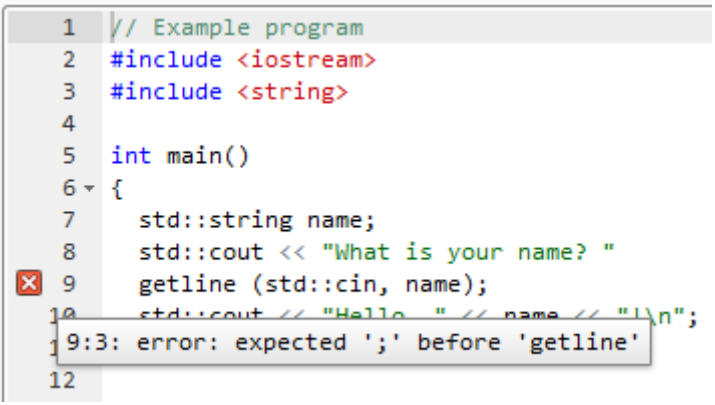
Jika percobaan penghapusan berhasil, maka sistem menampilkan pesan seperti pada Gambar 3.75.



Gambar 3.75 Antarmuka Percobaan Penghapusan Data Berhasil

3.6.2. Antarmuka Editor Teks

Antarmuka ini diimplementasikan pada editor teks yang ada. Editor teks ini dilengkapi dengan modul *instant feedback* yang mengenali kode program C++. Sehingga *syntax error* dan *code styling convention* yang ada, akan ditampilkan langsung ke dalam editor dalam bentuk error dan warning. Perancangan antarmuka ini bisa dilihat pada Gambar 3.76.



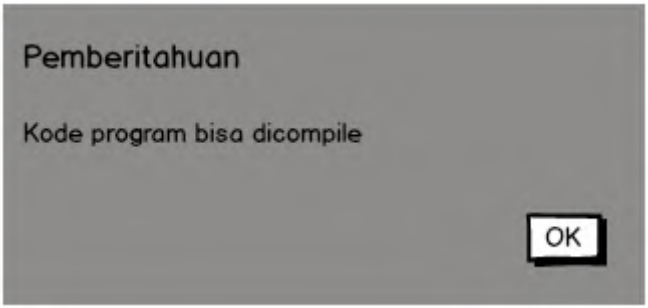
Gambar 3.76 Antarmuka Kode Editor dari cpp.sh

Karena proses pemberian umpan balik membutuhkan waktu beberapa saat, maka kode editor dilengkapi dengan Checker Status yang memiliki 2 *state* yaitu IDLE dan Checking.

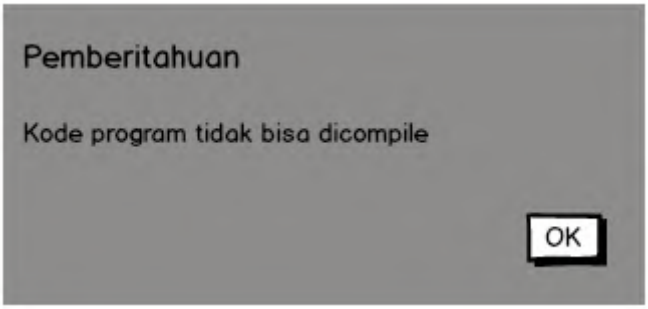
Selain itu, sistem ini juga bisa disambungkan dengan *compiler* lokal, yang *path*-nya bisa di-*update* di berkas .env. Saat melakukan proses *compile*, sistem akan memberikan tampilan Tunggu seperti yang terlihat pada Gambar 3.77.

Gambar 3.77 Antarmuka Menunggu Proses Compile

Setelah proses *compile* selesai, maka jika kode program yang ditulis bisa di-*compile*, maka akan muncul pemberitahuan seperti pada Gambar 3.78. Dan jika tidak, maka akan muncul pemberitahuan seperti pada Gambar 3.79.



Gambar 3.78 Antarmuka Pemberitahuan Kode Program Bisa Di-Compile



Gambar 3.79 Antarmuka Pemberitahuan Kode Program Tidak Bisa Di-Compile

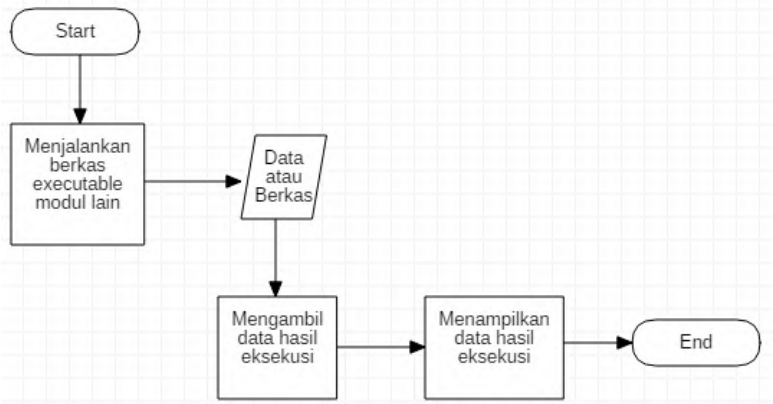
Jika kode program tidak bisa di-*compile*, sistem akan memberikan juga pemberitahuan dalam bentuk *error* ke dalam editor teks yang ada seperti pada Gambar 3.76.

3.7. Perancangan Proses Pemasangan Modul Lain

Platform elearning ini dirancang untuk bisa dimasuki atau diintegrasikan dengan modul lainnya. Untuk menangani hal ini, *platform elearning* harus memiliki 2 fungsi. Fungsi yang pertama adalah untuk melakukan proses eksekusi dari berkas *executable* dari modul lainnya. Dan fungsi yang kedua adalah untuk menampilkan hasil dari proses eksekusi di atas.

Selain itu, beberapa penyesuaian harus dilakukan. Penyesuaian di sini dapat berupa penyesuaian dalam basis data, *files*, struktur *folder*, dan lain sebagainya. Penyesuaian ini berguna untuk memastikan modul lain bisa terintegrasi dengan maksimal dengan *platform elearning* ini.

Pada , ditunjukkan bagaimana proses pemasangan modul lain agar bisa terintegrasi dengan *platform elearning* ini.



Gambar 3.80 Diagram Alir Proses Pemasangan Modul Lain

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1. Lingkungan Implementasi

Lingkungan implementasi yang akan digunakan untuk melakukan implementasi adalah sebagai berikut:

- Perangkat keras
  - Intel(R) Core(TM) i7-3517U CPU @ 1.90 GHz (4 CPUs), ~2.4 GHz
  - Memori RAM 4096 MB
- Perangkat lunak
  - Sistem Operasi: Windows Embedded 8.1 Industry Pro 64-bit (6.3, Build 9600)
  - Mozilla Firefox 45.0.1
  - Sublime Text Editor Build 3114
  - XAMPP Control Panel v3.2.2
  - PostgreSQL 9.4 dan pgAdmin III 1.20.0

4.2. Implementasi Antarmuka

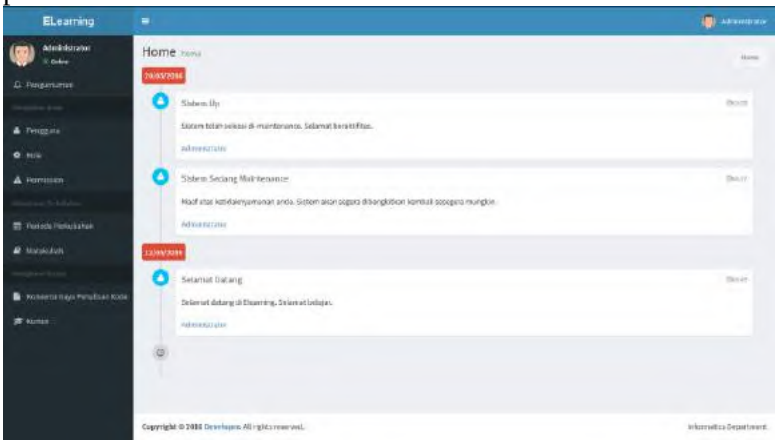
Pada subbab ini akan dijelaskan mengenai implementasi antarmuka dari aplikasi yang telah dibuat. Implementasi antarmuka dibagi menjadi 2 bagian yaitu *classroom management* dan *instant feedback system*.

4.2.1. Antarmuka *Classroom Management*

Classroom Management adalah modul di mana pengguna bisa melakukan proses manajemen elearning seperti *create*, *read*, *update*, dan *delete*.

4.2.1.1. Antarmuka *Timeline*

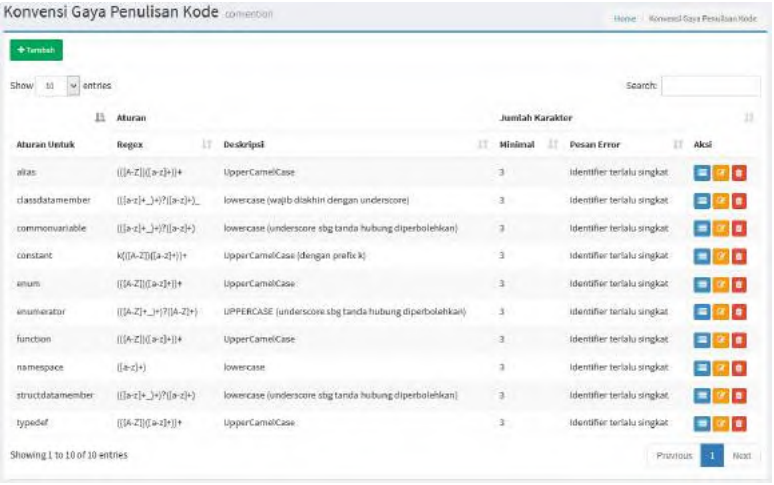
Antarmuka ini diimplementasikan pada semua *role* pengguna yang ada. Perbedaan yang ada hanya pada menu yang bisa diakses. Antarmuka ini dimunculkan pada saat pengguna telah berhasil melakukan *login*. Antarmuka *Timeline* dapat dilihat pada Gambar 4.1.



Gambar 4.1 Antarmuka *Timeline*

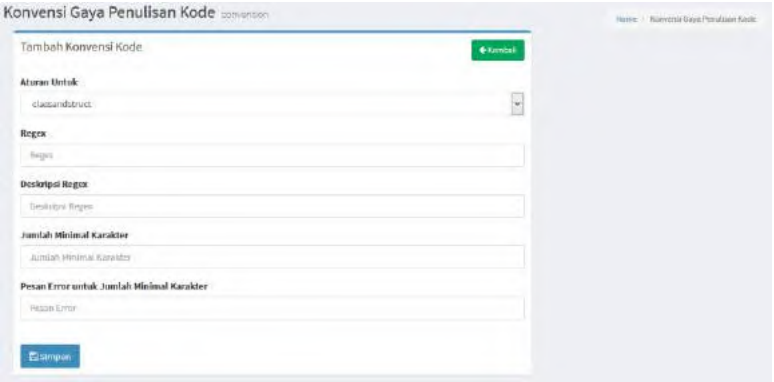
4.2.1.2. Antarmuka Manajemen Konvensi Gaya Penulisan Kode

Antarmuka ini diimplementasikan pada *role* Administrator. Halaman ini dapat diakses dengan memilih menu "Konvensi Gaya Penulisan Kode" yang ada pada navbar. Setelah pengguna memilih menu tersebut, muncullah tampilan seperti pada Gambar 4.2.

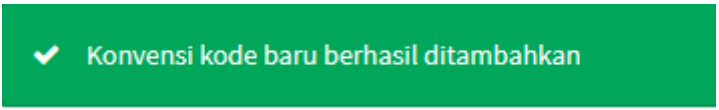


Gambar 4.2 Daftar Konvensi Gaya Penulisan Kode

Pengguna juga bisa menambahkan konvensi gaya penulisan kode baru. Pengguna bisa memilih tombol Tambah yang ada, untuk mengakses halaman seperti pada Gambar 4.3. Jika penambahan data berhasil, maka akan muncul pemberitahuan seperti pada Gambar 4.4.

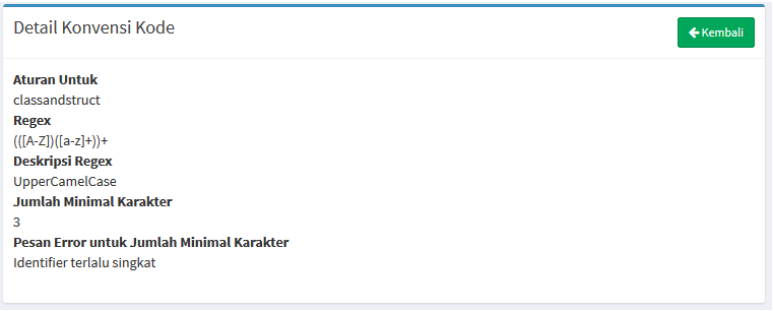


Gambar 4.3 Tambah Konvensi Gaya Penulisan Kode Baru



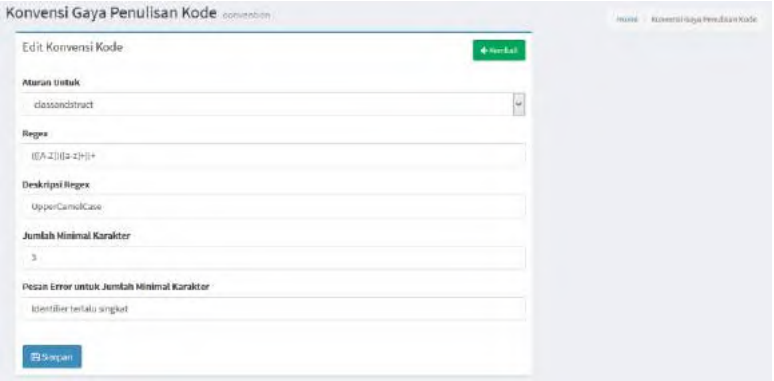
Gambar 4.4 Pemberitahuan Konvensi Gaya Penulisan Kode Baru Berhasil Ditambahkan

Pengguna bisa melihat detail dari konvensi gaya penulisan kode yang telah ada. Dengan memilih tombol Detail pada kolom Aksi, pengguna akan diarahkan ke halaman seperti pada Gambar 4.5.

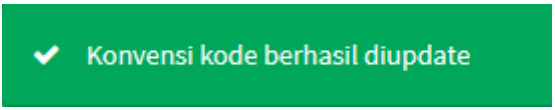


Gambar 4.5 Detail Konvensi Gaya Penulisan Kode

Pengguna juga bisa mengubah konvensi gaya penulisan kode yang telah ada. Dengan memilih tombol Edit pada kolom Aksi, pengguna akan diarahkan ke halaman seperti pada Gambar 4.6. Setelah berhasil melakukan pengubahan konvensi gaya penulisan kode, sistem akan memunculkan pemberitahuan seperti pada Gambar 4.7.

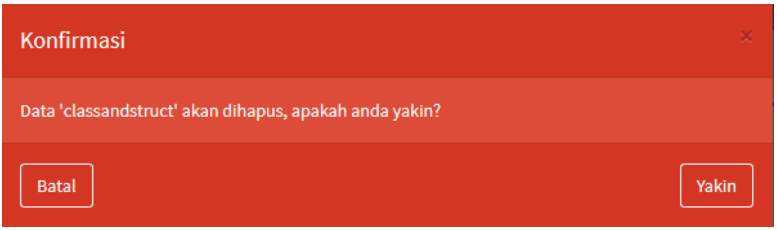


Gambar 4.6 Edit Konvensi Gaya Penulisan Kode



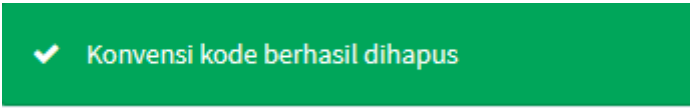
Gambar 4.7 Pemberitahuan Konvensi Gaya Penulisan Kode Berhasil Diedit

Pengguna juga bisa melakukan penghapusan pada konvensi gaya penulisan kode yang ada. Untuk melakukan aktivitas ini, pengguna memilih tombol Hapus yang ada pada kolom Aksi. Konfirmasi mengenai penghapusan data akan muncul dalam bentuk *modal* seperti pada Gambar 4.8.



Gambar 4.8 Konfirmasi Penghapusan Konvensi Gaya Penulisan Kode

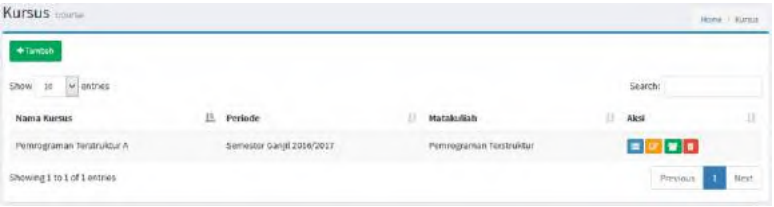
Jika pada saat konfirmasi penghapusan data pengguna memilih tombol Yakin, maka sistem akan melakukan percobaan penghapusan data. Karena pada umunya, data konvensi gaya penulisan kode tidak direferensikan oleh data lainnya, maka percobaan penghapusan data akan selalu berhasil dan sistem menampilkan pemberitahuan seperti pada Gambar 4.9.



Gambar 4.9 Pemberitahuan Konvensi Gaya Penulisan Kode Berhasil Dihapus

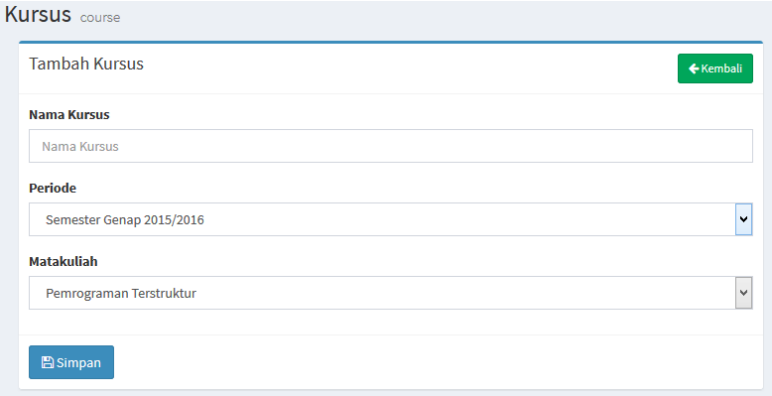
4.2.1.3. Antarmuka Manajemen Kursus

Antarmuka ini diimplementasikan pada *role* Administrator. Halaman ini dapat diakses dengan memilih menu "Kursus" yang ada pada navbar. Setelah pengguna memilih menu tersebut, muncullah tampilan seperti pada Gambar 4.10.

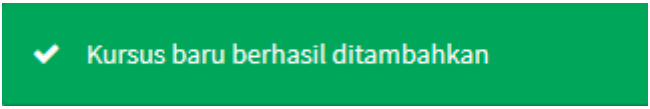


Gambar 4.10 Daftar Kursus

Pengguna juga bisa menambahkan kursus baru. Pengguna bisa memilih tombol Tambah yang ada, untuk mengakses halaman seperti pada Gambar 4.11. Jika penambahan data berhasil, maka akan muncul pemberitahuan seperti pada Gambar 4.12.

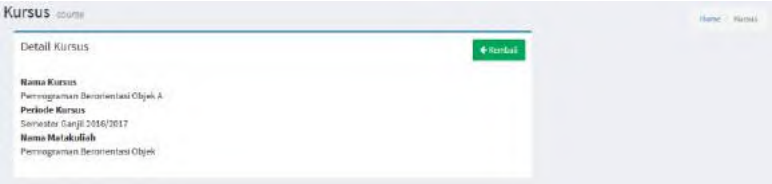


Gambar 4.11 Tambah Kursus Baru



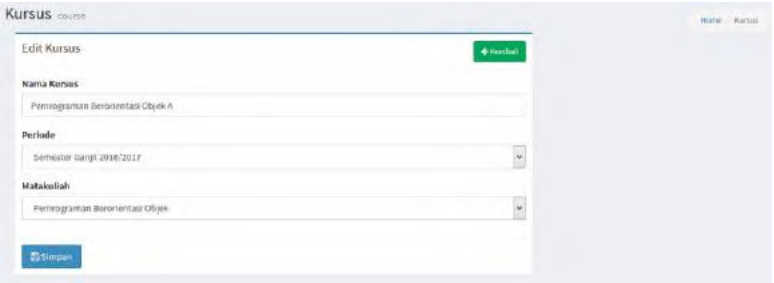
Gambar 4.12 Pemberitahuan Kursus Baru Berhasil Ditambahkan

Pengguna bisa melihat detail dari kursus yang telah ada. Dengan memilih tombol Detail pada kolom Aksi, pengguna akan diarahkan ke halaman seperti pada Gambar 4.13.

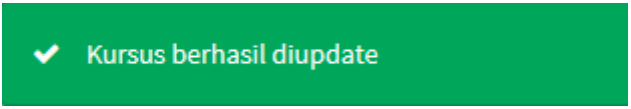


Gambar 4.13 Detail Kursus

Pengguna juga bisa mengubah kursus yang telah ada. Dengan memilih tombol Edit pada kolom Aksi, pengguna akan diarahkan ke halaman seperti pada Gambar 4.14. Setelah berhasil melakukan pengubahan kursus, sistem akan memunculkan pemberitahuan seperti pada Gambar 4.15.

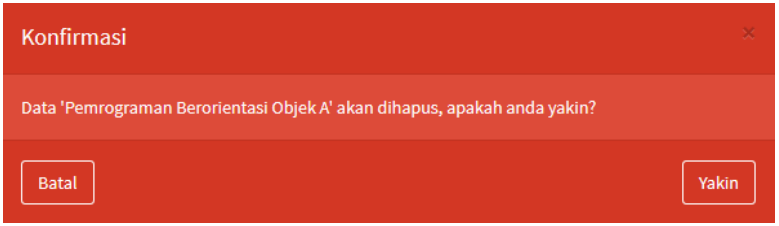


Gambar 4.14 Edit Kursus



Gambar 4.15 Pemberitahuan Kursus Berhasil Diedit

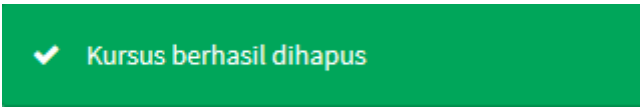
Pengguna juga bisa melakukan penghapusan pada kursus yang ada. Untuk melakukan aktivitas ini, pengguna memilih tombol Hapus yang ada pada kolom Aksi. Konfirmasi mengenai penghapusan data akan muncul dalam bentuk *modal* seperti pada Gambar 4.16.



Gambar 4.16 Konfirmasi Penghapusan Kursus

Jika pada saat konfirmasi penghapusan data pengguna memilih tombol Yakin, maka sistem akan melakukan percobaan penghapusan data. Jika percobaan penghapusan data berhasil, sistem akan menampilkan pemberitahuan seperti pada Gambar

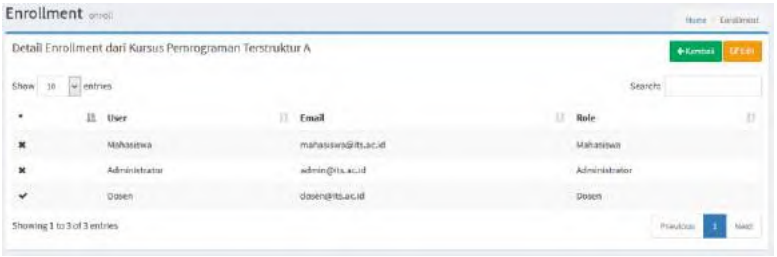
4.17. Jika data yang akan dihapus masih direferensikan oleh data lainnya, maka data gagal dihapus dan sistem akan menampilkan pemberitahuan seperti pada Gambar 4.18.



Gambar 4.17 Pemberitahuan Kursus Berhasil Dihapus

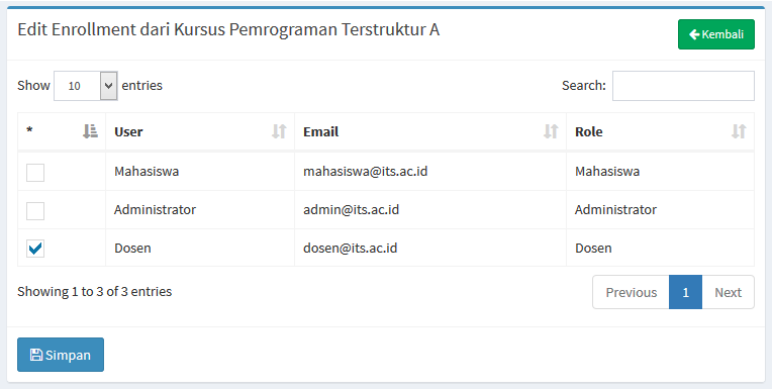
Gambar 4.18 Pemberitahuan Kursus Gagal Dihapus

Administrator bisa melakukan peng-*enroll*-an pengguna ke dalam kursus. Untuk melihat daftar pengguna yang belum atau telah *enroll*, administrator memilih tombol Enrollment pada kolom Aksi. Daftar pengguna yang telah *enroll* akan terlihat seperti pada Gambar 4.19.



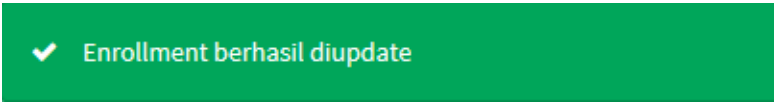
Gambar 4.19 Daftar *Enrollment* Pengguna

Administrator juga bisa mengedit siapa saja yang bisa *enroll* ke dalam kursus. Administrator bisa memilih tombol Edit, untuk menuju halaman seperti pada Gambar 4.20, untuk melakukan pengeditan.



Gambar 4.20 Edit *Enrollment* Pengguna

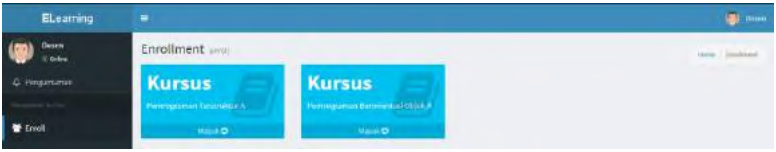
Setelah administrator memilih tombol Simpan, maka akan muncul pemberitahuan seperti pada Gambar 4.21.



Gambar 4.21 Pemberitahuan *Enrollment* Berhasil Diedit

4.2.1.4. Antarmuka Melihat Daftar Kursus

Antarmuka ini diimplementasikan pada *role* Dosen dan Mahasiswa. Antarmuka ini dimunculkan pada saat pengguna memilih menu "Enroll" di navbar. Antarmuka Melihat Daftar Kursus dapat dilihat pada Gambar 4.22.



Gambar 4.22 Antarmuka Melihat Daftar Kursus

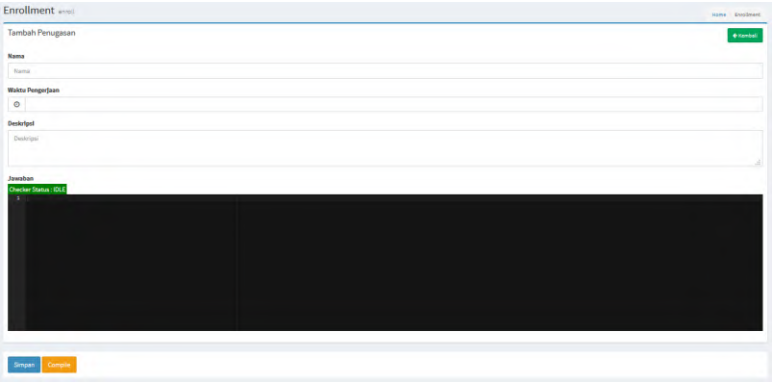
4.2.1.5. Antarmuka Manajemen Penugasan

Antarmuka ini diimplementasikan pada *role* Dosen dan Mahasiswa. *Role* Dosen memiliki hak akses penuh pada antarmuka ini. Tetapi *role* Mahasiswa hanya diberikan hak akses *read* saja. Halaman ini dapat diakses dengan memilih salah satu *box* yang ada di halaman daftar kursus. Setelah Dosen memilih *box* tersebut, muncullah tampilan seperti pada Gambar 4.23. Sedangkan pada Mahasiswa, tampilan akan menjadi seperti pada Gambar 4.24.

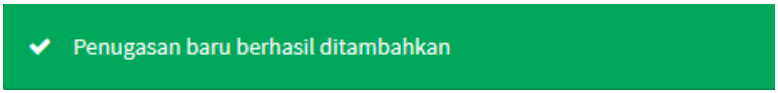
Gambar 4.23 Daftar Penugasan (Dosen)

Gambar 4.24 Daftar Penugasan (Mahasiswa)

Dosen juga bisa menambahkan penugasan baru. Dosen bisa memilih tombol Tambah yang ada, untuk mengakses halaman seperti pada Gambar 4.25. Jika penambahan data berhasil, maka akan muncul pemberitahuan seperti pada Gambar 4.26.

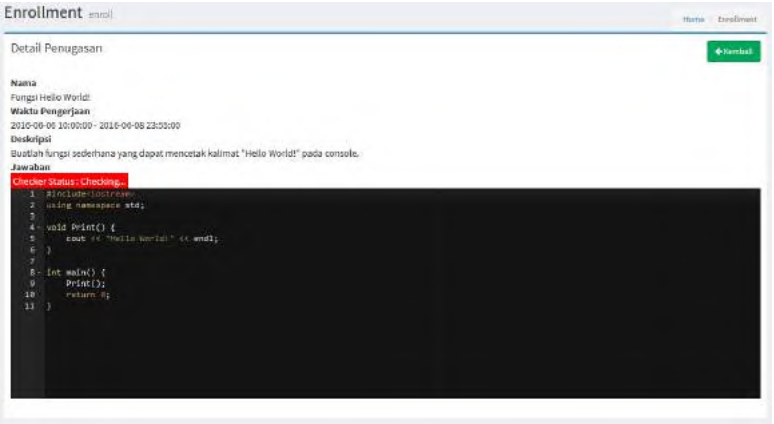


Gambar 4.25 Tambah Penugasan Baru



Gambar 4.26 Pemberitahuan Penugasan Baru Berhasil Ditambahkan

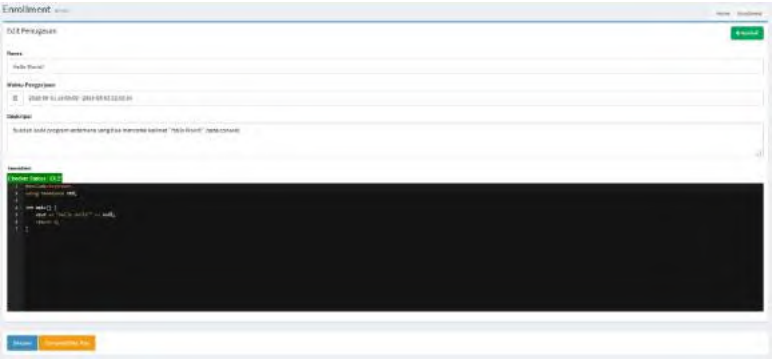
Pengguna bisa melihat detail dari penugasan yang telah ada. Dengan memilih tombol Detail pada kolom Aksi, Dosen akan diarahkan ke halaman seperti pada Gambar 4.27. Sedangkan Mahasiswa akan diarahkan ke halaman seperti pada Gambar 4.28.



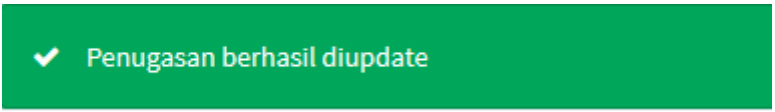
Gambar 4.27 Detail Penugasan (Dosen)

Gambar 4.28 Detail Penugasan (Mahasiswa)

Dosen juga bisa mengubah penugasan yang telah ada. Dengan memilih tombol Edit pada kolom Aksi, Dosen akan diarahkan ke halaman seperti pada Gambar 4.29. Setelah berhasil melakukan pengubahan penugasan, sistem akan memunculkan pemberitahuan seperti pada Gambar 4.30.

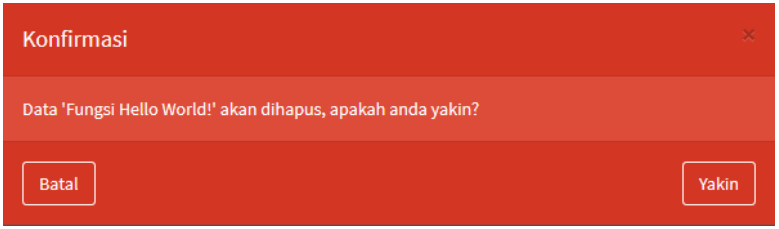


Gambar 4.29 Edit Penugasan



Gambar 4.30 Pemberitahuan Penugasan Berhasil Diedit

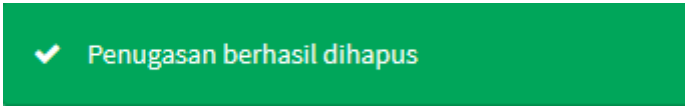
Dosen juga bisa melakukan penghapusan pada penugasan yang ada. Untuk melakukan aktivitas ini, Dosen memilih tombol Hapus yang ada pada kolom Aksi. Konfirmasi mengenai penghapusan data akan muncul dalam bentuk *modal* seperti pada Gambar 4.31.



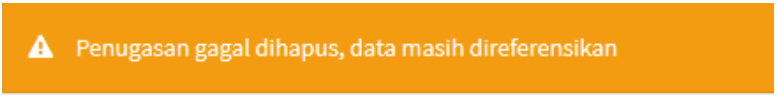
Gambar 4.31 Konfirmasi Penghapusan Penugasan

Jika pada saat konfirmasi penghapusan data Dosen memilih tombol Yakin, maka sistem akan melakukan percobaan penghapusan data. Jika percobaan penghapusan data berhasil,

sistem akan menampilkan pemberitahuan seperti pada Gambar 4.32. Jika data yang akan dihapus masih direferensikan oleh data lainnya, maka data gagal dihapus dan sistem akan menampilkan pemberitahuan seperti pada Gambar 4.33.



Gambar 4.32 Pemberitahuan Penugasan Berhasil Dihapus

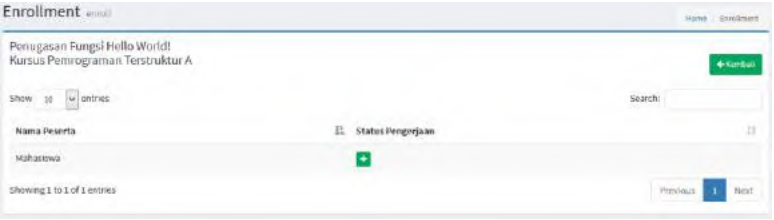


Gambar 4.33 Pemberitahuan Penugasan Gagal Dihapus

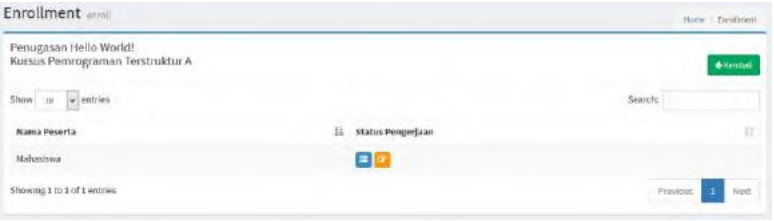
4.2.1.6. Antarmuka Manajemen Jawaban

Antarmuka ini diimplementasikan pada *role* Dosen dan Mahasiswa. Hak akses *delete*, tidak ada dalam antarmuka ini. *Role* Mahasiswa memiliki hak akses penuh pada antarmuka ini. Tetapi *role* Dosen hanya diberikan hak akses *read* saja. Halaman ini dapat diakses dengan memilih tombol Jawaban pada kolom Aksi di halaman Daftar Penugasan.

Setelah Mahasiswa memilih tombol tersebut, ada 2 kemungkinan tampilan yang akan muncul. Jika Mahasiswa belum pernah melakukan sesi menjawab penugasan, maka tampilan yang akan muncul adalah seperti pada Gambar 4.34. Sedangkan jika Mahasiswa pernah melakukan sesi menjawab penugasan, maka tampilan yang akan muncul adalah seperti pada Gambar 4.35. Perlu diperhatikan bahwa Mahasiswa hanya bisa melihat jawaban mereka sendiri.



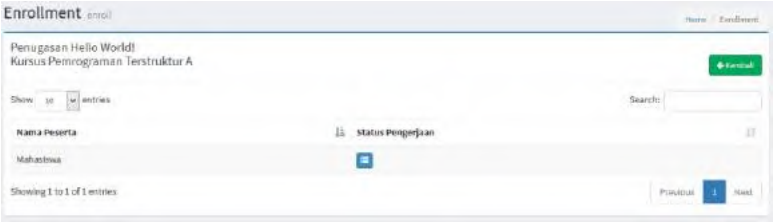
Gambar 4.34 Daftar Jawaban Jika Belum Pernah Melakukan Sesi Menjawab Penugasan (Mahasiswa)



Gambar 4.35 Daftar Jawaban Jika Sudah Pernah Melakukan Sesi Menjawab Penugasan (Mahasiswa)

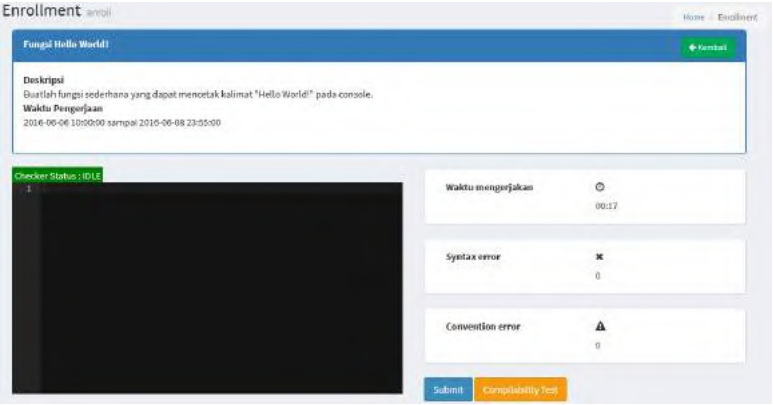
Lain halnya dengan Dosen. Dosen bisa melihat seluruh jawaban dari mahasiswa-mahasiswa yang *enroll* ke dalam kursus tersebut. Setelah Dosen memilih tombol tersebut, ada 2 kemungkinan tampilan yang akan muncul. Jika ada Mahasiswa yang belum pernah melakukan sesi menjawab penugasan, maka tampilan yang akan muncul adalah seperti pada Gambar 4.36. Sedangkan jika ada Mahasiswa yang pernah melakukan sesi menjawab penugasan, maka tampilan yang akan muncul adalah seperti pada Gambar 4.37.

Gambar 4.36 Daftar Jawaban Jika Mahasiswa Belum Pernah Melakukan Sesi Menjawab Penugasan (Dosen)

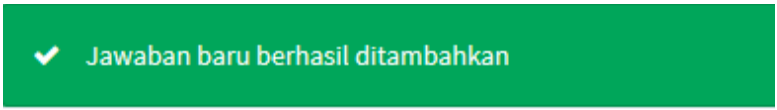


Gambar 4.37 Daftar Jawaban Jika Mahasiswa Sudah Pernah Melakukan Sesi Menjawab Penugasan (Dosen)

Mahasiswa bisa menambahkan jawaban baru selama belum pernah sama sekali melakukan sesi menjawab penugasan. Mahasiswa bisa memilih tombol Tambah Jawaban yang ada di kolom Status Pengerjaan, untuk mengakses halaman seperti pada Gambar 4.38. Jika penambahan data berhasil, maka akan muncul pemberitahuan seperti pada Gambar 4.39.

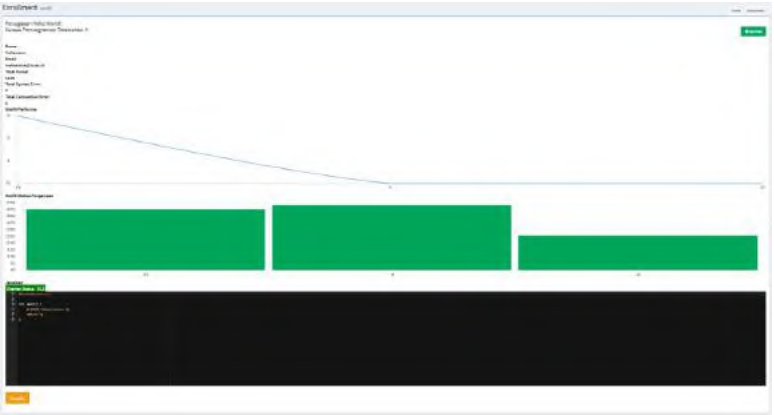


Gambar 4.38 Tambah Jawaban Baru



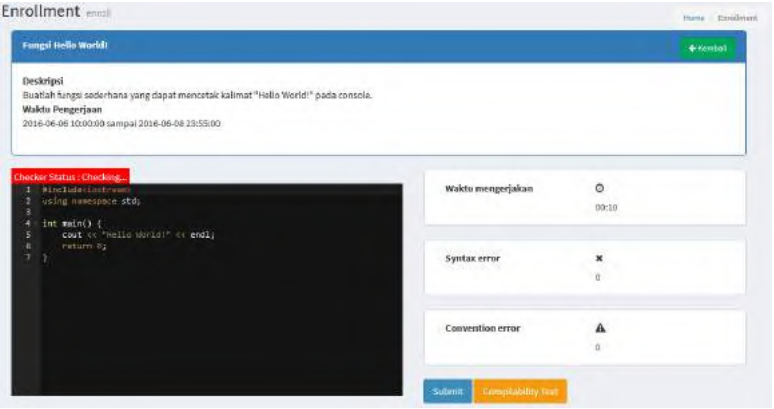
Gambar 4.39 Pemberitahuan Jawaban Baru Berhasil Ditambahkan

Dosen dan Mahasiswa bisa melihat detail dari jawaban yang telah ada. Dengan memilih tombol Lihat Jawaban pada kolom Status Pengerjaan, Dosen dan Mahasiswa akan diarahkan ke halaman seperti pada Gambar 4.40.



Gambar 4.40 Detail Jawaban (Dosen dan Mahasiswa)

Mahasiswa juga bisa mengubah jawaban yang telah ada. Dengan memilih tombol Edit Jawaban pada kolom Status Pengerjaan, Mahasiswa akan diarahkan ke halaman seperti pada Gambar 4.41. Setelah berhasil melakukan pengubahan jawaban, sistem akan memunculkan pemberitahuan seperti pada Gambar 4.42.



Gambar 4.41 Edit Jawaban

Gambar 4.42 Pemberitahuan Jawaban Berhasil Diedit

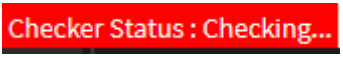
4.2.2. Antarmuka *Instant Feedback System*

Antarmuka ini diterapkan pada editor teks yang ada yaitu Ace Editor. Ace Editor memiliki fitur-fitur yang cukup lengkap untuk mendukung penulisan bahasa pemrograman yang ada. Untuk *platform elearning* ini, *mode* yang digunakan adalah *mode* C++. Sehingga nantinya editor teks akan mendukung *Syntax Highlighting* untuk kode program C++. *Syntax Highlighting* yang sudah diterapkan bisa dilihat pada Gambar 4.43.

Gambar 4.43 *Syntax Highlighting*

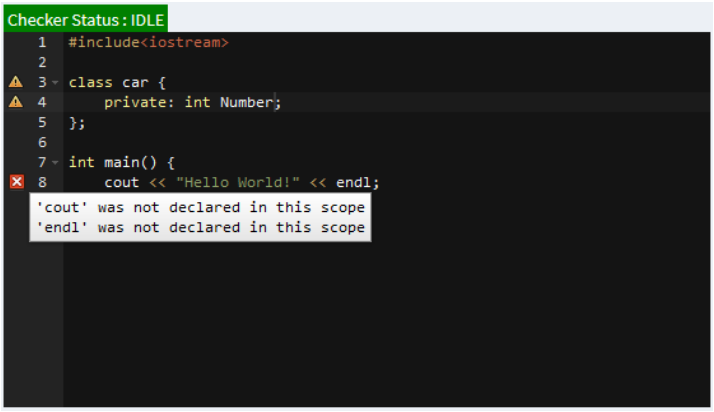
Proses pemberian umpan balik pada editor teks tentunya memakan waktu beberapa saat. Oleh karena itu, di tepi kiri atas editor teks, ditampilkan status dari *checker*. Jika *checker* sedang dalam keadaan *idle*, maka tampilan akan seperti pada Gambar 4.44. Sedangkan apabila *checker* sedang dalam keadaan sibuk, maka tampilan akan seperti pada Gambar 4.45.

Gambar 4.44 *Checker* sedang dalam keadaan *idle*

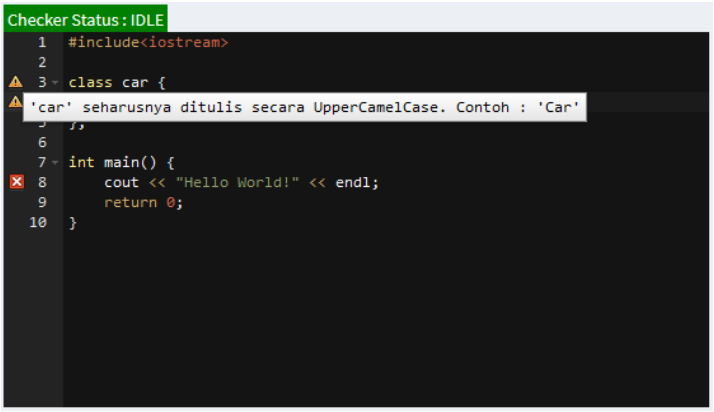


Gambar 4.45 *Checker* sedang dalam keadaan sibuk

Setelah proses pengecekan selesai, jika ada kesalahan dalam *syntax*, maka di editor teks akan muncul tanda *error*. Jika tanda tersebut di-*hover*, maka akan muncul pesan seperi pada Gambar 4.46. Dan jika ada kesalahan dalam *styling*, maka di editor akan muncul tanda *warning*. Jika tanda tersebut di-*hover*, maka akan muncul pesan dan sugesti seperi pada Gambar 4.47.

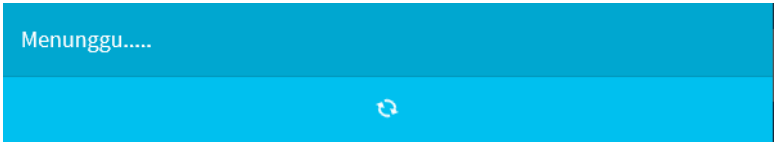


Gambar 4.46 Pesan Saat Terjadi *Syntax Error*



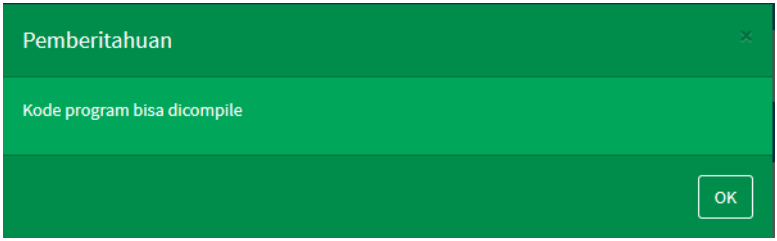
Gambar 4.47 Pesan dan Sugesti Saat Terjadi *Code Styling Convention Error*

Fitur lain yang ada dalam *platform elearning* ini yang masih ada kaitannya dengan penulisan kode program adalah fitur *compile*. Dosen dan Mahasiswa yang ingin melakukan pengecekan kode program menggunakan *compiler*, bisa memilih tombol Compile yang ada. Jika kompilasi sedang dalam proses, sistem akan memberikan pesan seperti pada Gambar 4.48.

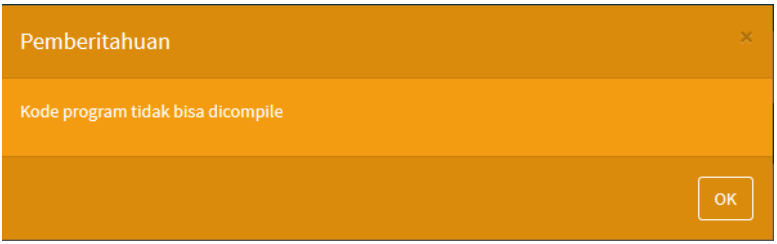


Gambar 4.48 *Compiler* Sedang Melakukan Kompilasi

Setelah *compiler* selesai mengompilasi kode program, akan ada 2 kemungkinan tampilan yang akan muncul, yaitu berhasil dan gagal. Jika berhasil, maka akan muncul pesan seperti pada Gambar 4.49. Dan jika gagal, akan muncul pesan seperti pada Gambar 4.50. Selain memunculkan pesan gagal dalam bentuk *modal*, pada editor teks juga akan dimunculkan tanda *error* yang menunjukkan detail kesalahan apa yang terjadi sehingga menyebabkan kegagalan dalam kompilasi, seperti yang terlihat pada Gambar 4.51.



Gambar 4.49 Kompilasi Kode Program Berhasil



Gambar 4.50 Kompilasi Kode Program Gagal

Gambar 4.51 Detail Penyebab Kegagalan Proses Kompilasi

4.3. Implementasi Fungsionalitas

Implementasi fungsionalitas ini berdasarkan pada kasus penggunaan yang ada di dalam aplikasi. Secara penulisan kode sumber program, implementasi ini dibagi menjadi:

1. Manajemen Konvensi Gaya Penulisan Kode, yang mencakup
    a. UC-01
    b. UC-02
    c. UC-03

   d. UC-04
   e. UC-05
2. Manajemen Kursus, yang mencakup
   a. UC-06
   b. UC-07
   c. UC-08
   d. UC-09
   e. UC-10
3. Manajemen *Enrollment*, yang mencakup
   a. UC-11
   b. UC-13
4. Manajemen Penugasan, yang mencakup
   a. UC-14
   b. UC-15
   c. UC-16
   d. UC-17
   e. UC-18
5. Manajemen Jawaban, yang mencakup
   a. UC-19
   b. UC-20
   c. UC-21
   d. UC-22

6. *Instant Feedback System*, yang mencakup
   a. UC-23
   b. UC-24

7. Sistem *Timeline* dan *Compiler*, yang mencakup
   a. UC-12
   b. UC-24

Laravel memiliki *style*-nya sendiri dalam menangani kasus CRUD seperti kasus manajemen data dan basis data. Laravel memiliki *Resource Controllers* untuk menangani hal ini. Untuk subbab 4.3.1, 4.3.2, 4.3.3, 4.3.4, dan 4.3.5 akan menggunakan

pendekatan ini. *Controller* yang di-*generate* oleh Laravel memiliki *function-function* deskripsinya dijelaskan pada Tabel 4.1.

Tabel 4.1 Deskripsi Fungsi-Fungsi *Resource Controllers*

No	Nama Fungsi	Parameter (default)	Kegunaan
1.	index		Untuk menampilkan daftar *resource* yang tersimpan di dalam basis data.
2.	create		Untuk menampilkan *form* yang digunakan untuk membuat *resource* baru.
3.	store	Request $request	Untuk menyimpan *resource* baru ke dalam basis data
4.	show	$id	Untuk menampilkan *resource* secara spesifik yang tersimpan di dalam basis data
5.	edit	$id	Untuk menampilkan *resource* yang ada di dalam basis data dan menampilkannya ke dalam *form* yang digunakan untuk mengedit *resource*.
6.	update	Request $request, $id	Untuk memperbarui *resource* yang sebelumnya telah tersimpan di dalam basis data
7.	destroy	$id	Untuk menghapus *resource* yang dipilih dari basis data.

4.3.1. Manajemen Konvensi Gaya Penulisan Kode

Di dalam proses manajemen gaya penulisan kode, terdapat implementasi kode program untuk mendapatkan, memasukkan, mengubah, dan menghapus data konvensi gaya penulisan kode dari basis data. Kelas yang mengatur aktivitas ini adalah kelas ConventionController yang dapat dilihat Kode Sumber 7.1 ada di LAMPIRAN A – KODE SUMBER CONTROLLER.

Karena proses pengoreksian *code styling convention* membutuhkan pemanggilan fungsi ada di dalam *Listener*, maka perlu adanya pendekatan *hardcode* pada sistem ini. Maka di konfigurasi dari Laravel ini ditambahkan daftar-daftar *type identifier* apa saja yang bisa / telah di-*cover*. Pada Kode Sumber 4.1, ditunjukkan daftar dari *type identifier* apa saja yang bisa / telah di-*cover* oleh sistem.

4.3.2. Manajemen Kursus

Di dalam proses manajemen kursus, terdapat implementasi kode program untuk mendapatkan, memasukkan, mengubah, dan menghapus data kursus dari basis data. Kelas yang mengatur aktivitas ini adalah kelas CourseController yang dapat dilihat pada Kode Sumber 7.2 yang ada di LAMPIRAN A – KODE SUMBER CONTROLLER.

4.3.3. Manajemen *Enrollment*

Di dalam proses manajemen *enrollment*, terdapat implementasi kode program untuk mendapatkan, memasukkan, mengubah, dan menghapus data *enrollment* dari basis data. Kelas yang mengatur aktivitas ini adalah kelas EnrollController yang dapat dilihat pada Kode Sumber 7.3 yang ada di LAMPIRAN A – KODE SUMBER CONTROLLER.

```
return [
        'all' => [
                0 => 'classandstruct',
                1 => 'typedef',
                2 => 'alias',
                3 => 'enum',
                4 => 'commonvariable',
                5 => 'classdatamember',
                6 => 'structdatamember',
                7 => 'constant',
                8 => 'function',
                9 => 'enumerator',
                10 => 'namespace',
        ],
        'classandstruct' => 'classandstruct',
        'typedef' => 'typedef',
        'alias' => 'alias',
        'enum' => 'enum',
        'commonvariable' => 'commonvariable',
        'classdatamember' => 'classdatamember',
        'structdatamember' => 'structdatamember',
        'constant' => 'constant',
        'function' => 'function',
        'enumerator' => 'enumerator',
        'namespace' => 'namespace',
];
```

Kode Sumber 4.1 Daftar *Type Identifier*

4.3.4. Manajemen Penugasan

Di dalam proses manajemen penugasan, terdapat implementasi kode program untuk mendapatkan, memasukkan, mengubah, dan menghapus data penugasan dari basis data. Kelas yang mengatur aktivitas ini adalah kelas QuizController yang dapat dilihat pada Kode Sumber 7.4 yang ada di LAMPIRAN A – KODE SUMBER CONTROLLER.

4.3.5. Manajemen Jawaban

Di dalam proses manajemen jawaban, terdapat implementasi kode program untuk mendapatkan, memasukkan, dan mengubah data jawaban dari basis data. Kelas yang mengatur aktivitas ini adalah kelas AnswerController yang dapat dilihat pada Kode Sumber 7.5 yang ada di LAMPIRAN A – KODE SUMBER CONTROLLER.

4.3.6. *Instant Feedback System*

Untuk mendukung proses penulisan kode program, dibutuhkan suatu editor teks berbasis *web*. Editor teks berbasis web pada *platform elearning* ini adalah Ace Editor. Pada Kode Sumber 4.2 ditunjukkan cara menyematkan Ace Editor ke dalam *file* yang ada.

```
editor = ace.edit("editor");
editor.setTheme("ace/theme/twilight");
editor.getSession().setMode("ace/mode/c_cpp");
```

Kode Sumber 4.2 Menyematkan Ace Editor ke dalam Sistem

Karena diperlukannya suatu *web worker* untuk mengatasi *delay* pada saat pemrosesan kode program menjadi AST, maka perlu adanya pendeklarasian *web worker* pada Ace Editor, seperti yang ditunjukkan pada Kode Sumber 4.3.

```
this.createWorker = function(session) {
    var worker = new WorkerClient(["ace"], "ace/worker/my-
        worker", "MyWorker", http://localhost/plagiarism-
        checker/Projects/elearning/public/js/my-worker.js ");
    worker.attachToDocument(session.getDocument());
}
```

Kode Sumber 4.3 Membuat *Web Worker* pada Ace Editor

Web worker yang ada akan mengacu pada *web worker* standar milik Ace Editor. Selain itu, *web worker* juga harus me-

load file ANTLR Javascript Target berserta dengan *grammar* yang diinginkan. Untuk implementasinya dapat dilihat pada Kode Sumber 4.4.

```
importScripts("worker-base.js");

// load nodejs compatible require
var ace_require = require;
require = undefined;
var Honey = { 'requirePath': ['..'] }; // walk up to js folder, see Honey
        docs
importScripts("require.js");
var antlr4_require = require;
require = ace_require;

// load antlr4 and myLanguage
var antlr4, mylanguage;
try {
    require = antlr4_require;
    antlr4 = require('js/antlr4/index');
    var CPP14Lexer = require('js/cpp/CPP14Lexer');
    var CPP14Parser = require('js/cpp/CPP14Parser');
    var CPP14Listener = require('js/cpp/CPP14Listener');
    var CPP14Visitor = require('js/cpp/CPP14Visitor');
} finally {
    require = ace_require;
}
```

Kode Sumber 4.4 Inisialisasi ANTLR Javascript Target

Validasi *syntax* dan *code styling convention* akan selalu dilakukan setiap pengguna melakukan proses pengetikan kode program. Implementasi pemanggilan fungsi validasi pada sistem adalah seperti pada Kode Sumber 4.5. Dan untuk pemanggilan fungsi tersebut, akan di-*cover* oleh sistem, yaitu pada Kode Sumber 4.6.

```
var validate = function(input) {
    var stream = new antlr4.InputStream(input);
```

```
    var lexer = new CPP14Lexer.CPP14Lexer(stream);
    var tokens = new antlr4.CommonTokenStream(lexer);
    var parser = new CPP14Parser.CPP14Parser(tokens);

    var annotations = [];
    var listener = new AnnotatingErrorListener(annotations)
    parser.removeErrorListeners();
    parser.addErrorListener(listener);
    var tree = parser.translationunit();
    parser.buildParseTrees = true;

    var printer = new StatementPrinter(annotations);
    antlr4.tree.ParseTreeWalker.DEFAULT.walk(printer, tree);

    return annotations;
};
```

Kode Sumber 4.5 Fungsi Validasi AST

```
    (function() {
        this.onUpdate = function() {
            var value = this.doc.getValue();
            var annotations = validate(value);
            this.sender.emit("annotate", annotations);
        };

    }).call(MyWorker.prototype);
```

Kode Sumber 4.6 Fungsi onUpdate Yang Memanggil Fungsi Validasi Setiap Terjadi Perubahan Isi Editor

Pengecekan yang pertama kali dilakukan oleh *listener* adalah *syntax* yang ada. Dalam Kode Sumber 4.5, dijelaskan bahwa objek dari kelas AnnotatingErrorListener bertanggung jawab untuk menangani kasus ini. Pada Kode Sumber 4.7, dijelaskan deklarasi dari kelas AnnotatingErrorListener. Jika terjadi *syntax error*, kelas ini bertanggung jawab untuk menyusun pesan error dan memasukkannya ke dalam Ace Editor.

```
// class for gathering errors and posting them to ACE editor
var AnnotatingErrorListener = function(annotations) {
```

```
    antlr4.error.ErrorListener.call(this);
    this.annotations = annotations;
    return this;
};

AnnotatingErrorListener.prototype =
Object.create(antlr4.error.ErrorListener.prototype);
AnnotatingErrorListener.prototype.constructor =
AnnotatingErrorListener;

AnnotatingErrorListener.prototype.syntaxError =
function(recognizer, offendingSymbol, line, column, msg, e) {
    this.annotations.push({
        row: line - 1,
        column: column,
        text: msg,
        type: "error"
    });
};
```

Kode Sumber 4.7 Kelas AnnotatingErrorListener untuk Menangani *Syntax Error*

Setelah dipastikan tidak mengalami *syntax error*, *listener* selanjutnya bertanggung jawab untuk memastikan *code styling convention* yang ada sudah sesuai dengan aturan yang telah ditetapkan. Pada Kode Sumber 4.5, telah dijelaskan bahwa objek yang bertanggung jawab adalah objek dari kelas StatementPrinter. Deklarasi dari kelas StatementPrinter adalah seperti pada Kode Sumber 4.8.

```
StatementPrinter = function(annotations) {
    CPP14Listener.CPP14Listener.call(this);
    this.annotations = annotations;
    return this;
};

StatementPrinter.prototype =
Object.create(CPP14Listener.CPP14Listener.prototype);
```

```
StatementPrinter.prototype.constructor = StatementPrinter;
```

Kode Sumber 4.8 Kelas StatementPrinter untuk Melakukan Penelusuran Pada AST

Kelas StatementPrinter adalah turunan dari kelas CPP14Listener yang memiliki beberapa *method* untuk menelusuri *node-node* yang ada dalam AST. Untuk method yang digunakan oleh sistem ini, dapat dilihat pada Kode Sumber 8.1 yang ada di LAMPIRAN B – METHOD-METHOD LISTENER. Setiap *method* tersebut akan memanggil fungsi ConventionCheck untuk dilakukan pengecekan *Code Styling Convention*. Implementasi dari fungsi ini ada pada Kode Sumber 4.9.

Untuk mendapatkan *code styling convention* yang berbentuk RegEx, fungsi ConventionCheck di atas melakukan pengambilan data RegEx sesuai dengan kebutuhan *type identifier* yang ditemukan. Implementasi dari pengambilan RegEx adalah seperti pada Kode Sumber 4.10.

```
var ConventionCheck = function(sfor, ctx, annotations, textspecial) {
    var retext = GetRegex(sfor);
    if(retext != null) {
        var re = new RegExp(retext, 'g');
        if(textspecial) {
            var text = textspecial;
        } else {
            var text = ctx.getText();
        }
        GetMinLength(sfor, text, ctx, annotations);
        var stringexec = re.exec(text);
        if(stringexec == null || text != stringexec[0]) {
            var convmessage = GetConventionMessage(sfor);
            if(convmessage == null) {
                convmessage = "{deskripsi aturan tidak ditemukan}"
            }
            var suggest = SuggestionText(sfor, retext, text, stringexec);
            annotations.push({
                row: ctx.start.line - 1,
```

```
            column: ctx.start.column,
            text: """ + text + "' seharusnya ditulis secara " +
          convmessage + ". Contoh : '" + suggest + """,
            type: "warning"
          });
      }
   }
}
```

Kode Sumber 4.9 Fungsi ConventionCheck untuk Mengecek *Code Styling Convention*

```
var GetRegex = function(sfor) {
   var request = new XMLHttpRequest();
   request.open('GET', "http://localhost/plagiarism-
         checker/Projects/elearning/public/convention/getregex/" +
         sfor, false);  // `false` makes the request synchronous
   request.send(null);

   if (request.status === 200) {
      response = JSON.parse(request.responseText);
      if(response.length == 0) {
         return null;
      } else {
         regex = response[0].regex;
         return regex;
      }
   } else {
      return null;
   }
}
```

Kode Sumber 4.10 Pengambilan Data RegEx

Selain RegEx, fungsi ConventionCheck juga mengambil data tentang panjang minimal suatu *identifier* dan pesan *error* apabila panjang minimal tidak terpenuhi. Implementasi dari pengambilan panjang minimal suatu *identifier* dan pesan *error*-nya adalah seperti yang ditulis pada Kode Sumber 4.11.

```
var GetMinLength = function(sfor, text, ctx, annotations) {
    var request = new XMLHttpRequest();
    request.open('GET', "http://localhost/plagiarism-
            checker/Projects/elearning/public/convention/getconvmin/"
            + sfor, false);  // `false` makes the request synchronous
    request.send(null);

    if (request.status === 200) {
        response = JSON.parse(request.responseText);
        if(response.length > 0) {
            min = response[0].min;
            if(text.length < min) {
                annotations.push({
                    row: ctx.start.line - 1,
                    column: ctx.start.column,
                    text: response[0].pesanmin + ", minimal " + min + "
                karakter",
                    type: "warning"
                });
            }
        }
    }
}
```

Kode Sumber 4.11 Pengambilan Data Panjang Minimal Suatu *Identifier* dan Pesan *Error*-nya

Jika ditemukan adanya *code styling convention error*, maka fungsi ConventionCheck akan memanggil fungsi GetConventionMessage yang bertugas untuk mengambil deskripsi dari *code styling convention* yang telah diatur oleh Administrator. Implementasi dari fungsi GetConventionMessage adalah seperti pada Kode Sumber 4.12.

```
var GetConventionMessage = function(sfor) {
   var request = new XMLHttpRequest();
   request.open('GET', "http://localhost/plagiarism-
         checker/Projects/elearning/public/convention/getconvmessa
         ge/" + sfor, false);  // `false` makes the request synchronous
   request.send(null);

   if (request.status === 200) {
      response = JSON.parse(request.responseText);
      if(response.length == 0) {
         return null;
      } else {
         deskripsi = response[0].deskripsi;
         return deskripsi;
      }
   } else {
      return null;
   }
}
```

Kode Sumber 4.12 Pengambilan Deskripsi *Code Styling Convention*

```
var SuggestionText = function(sfor, retext, text, stringexec) {
   var re = new RegExp(retext, 'g');
   if(sfor == 'commonvariable' || sfor == 'structdatamember') {
      var striped = text.replace(/_+/gi, '_');
      var lower = striped.toLowerCase();
      var suggest = re.exec(lower);
      return suggest[0];
   } else if(sfor == 'classdatamember') {
      var added = text + '_';
      var striped = added.replace(/_+/gi, '_');
      var lower = striped.toLowerCase();
      var suggest = re.exec(lower);
      return suggest[0];
   } else if(sfor == 'namespace') {
      var striped = text.replace(/_+/gi, '');
      var suggest = striped.toLowerCase();
      return suggest;
```

```
    } else if(sfor == 'enumerator') {
        var striped = text.replace(/_+/gi, '_');
        var upper = striped.toUpperCase();
        var suggest = re.exec(upper);
        return suggest[0];
    } else if(sfor == 'globalvariable') {
        var striped = text.replace(/_+/gi, '_');
        var lower = striped.toLowerCase();
        var final = re.exec(lower);
        if(final == null) {
            var suggest = 'g_' + final[0];
        } else {
            var suggest = final[0];
        }
        return suggest;
    } else if(sfor == 'classandstruct' || sfor == 'typedef' || sfor == 'alias' ||
sfor == 'enum' || sfor == 'function' || sfor == 'constant') {
        var suggest = '';
        if(text.indexOf('_') > -1) {
            var striped = text.replace(/_+/gi, '_');
            var arrsplit = striped.split('_');
            arrsplit.forEach(function(splited) {
                suggest += UpperCamelCase(splited, retext);
            });
        } else {
            suggest = UpperCamelCase(text, retext);
        }

        if(sfor == 'constant') {
            return 'k' + suggest;
        } else {
            return suggest;
        }
    }
}
```

Kode Sumber 4.13 Pemberian Sugesti Penulisan Kode Program yang Sesuai dengan Code Styling Convention

Setelah itu fungsi ConventionCheck memanggil fungsi SuggestionText untuk memberikan sugesti terkait dengan penulisan kode program benar sesuai dengan *code styling convention*. Implementasi dari SuggestionText adalah seperti pada Kode Sumber 4.13.

Setelah mendapatkan sugesti, maka fungsi ConventionCheck akan menyusun pesan *error* dan mengirimkannya ke Ace Editor untuk dijadikan sebagai *warning*. Pengecekan ini akan memakan waktu beberapa saat, tergantung pada banyaknya *line of code*.

4.3.7. Sistem *Timeline* dan *Compiler*

Tampilan berupa *timeline* dimunculkan ke pengguna setelah pengguna berhasil melakukan *login* atau pengguna menuju halaman *home.* Selain itu, *platform elearning* ini dilengkapi dengan fitur yang mampu melakukan pengetesan tentang bisa di-*compile* atau tidaknya suatu kode program. Semua fitur di atas diimplementasikan pada satu kelas yang bernama HomeController dengan fungsi bernama index dan compile. Implementasi dari kelas ini dapat dilihat pada Kode Sumber 4.15. Untuk mendefinisikan lokasi dari *compiler, server administrator* harus men-*set* lokasi compiler yang ada di file .env seperti contoh pada Kode Sumber 4.14.

```
COMPILER='C:\"Program Files (x86)"\Dev-Cpp\MinGW64\bin\g++'
```

Kode Sumber 4.14 Contoh Lokasi *Compiler* yang Ditulis di .env

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests;
use Illuminate\Http\Request;
use App\Post;
use DB;
use Input;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $posts = DB::table('elearning.posting')
        ->leftJoin('users', 'users.id', '=', 'elearning.posting.user_id')
        ->select('elearning.posting.*', 'users.name')
        ->orderBy('elearning.posting.created_at', 'desc')
        ->paginate(3);

        return view('home', ['posts' => $posts]);
    }
```

```
    public function compile(Request $request) {
        if($request->ajax()){
            $file = fopen("test.cpp", 'w');
            fwrite($file, $request->source);
            fclose($file);

            exec(trim(env('COMPILER'),"") . ' 2>&1 test.cpp -O3 -o
              test.exe', $res, $ret);

            return $res;
        }

    }
}
```

Kode Sumber 4.15 Kelas HomeController

4.4. Implementasi Pemasangan Modul *Student Feedback System*

Untuk melakukan pemasangan modul *Student Feedback System* ini, yang pertama kali dilakukan adalah mengeksekusi *file executable* yang telah dimasukkan ke berkas-berkas *platform elearning*. Fungsi yang bertugas untuk melakukan proses ini adalah fungsi check, yang dijelaskan pada Kode Sumber 4.16.

```
public function check(Request $request) {
            chdir('similarity_plugin');
            $inputfile = fopen('Path/pathFolderSourceCode.txt', 'w');
            $input = "input\\" . "\r\n" . $request->courseid . "\r\n" .
$request->quizid . "\r\n" . "output\\";
            fwrite($inputfile, $input);
            fclose($inputfile);
            exec("java -jar " . trim(env('SIMILARITY'),""));
            chdir('..');
            return "true";
}
```

Kode Sumber 4.16 Mengeksekusi *File Executable* dari Modul *Student Feedback System*

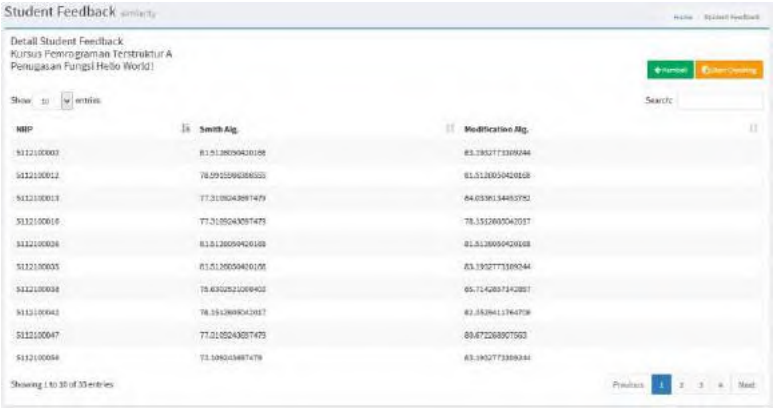
```
public function index($courseid, $quizid) {
        $course = Course::find($courseid);
        $quiz = Quiz::find($quizid);

        $similaritys = DB::table('sc_similarity')
        ->select('*')
        ->where('kelas', '=', $courseid)
        ->where('kodesoal', '=', $quizid)
        ->get();

        return view('similarity.index', ['courseid' => $courseid, 'quizid' => $quizid, 'similaritys' => $similaritys, 'course' => $course, 'quiz' => $quiz]);
}
```

Kode Sumber 4.17 Mengambil Data dan Menampilkan *Student Feedback*

Setelah dilakukan proses eksekusi, maka selanjutnya adalah mengambil data hasil eksekusi yang tersimpan di dalam basis data dan menampilkannya ke halaman *web*. Fungsi index seperti yang ditunjukkan pada Kode Sumber 4.17, bertugas untuk melakukan proses ini.

Antarmuka dari *Student Feedback System* akan dipasang pada *role* Dosen. Sehingga pengguna dengan *role* Dosen bisa dengan mudah melakukan pengecekan pekerjaan mahasiswa. Pada Gambar 4.52 ditunjukkan bagaimana hasil proses pengecekan yang dilakukan, dengan kolom yang menunjukkan berapa persentase kemiripan pekerjaan mahasiswa dengan kunci jawaban dosen.

Untuk penjelasan lebih jauh mengenai proses pengecekan *similarity* ini, pembaca bisa merujuk ke tugas akhir yang berjudul Rancang Bangun E-Learning Pemrograman pada Modul Student Feedback System yang disusun oleh saudari Rachmania Ilavi NRP 5112100168 [10].



Gambar 4.52 Antarmuka Modul *Student Feedback System*

4.5. Implementasi Pemasangan Modul *Plagiarism Detection System*

Untuk melakukan pemasangan modul *Plagiarism Detection System* ini, yang pertama kali dilakukan adalah mengeksekusi *file executable* yang telah dimasukkan ke berkas-berkas *platform elearning*. Fungsi yang bertugas untuk melakukan proses ini adalah fungsi check, yang dijelaskan pada Kode Sumber 4.18.

```
public function check(Request $request) {
        chdir('plagiarism_plugin');
        $inputfile = fopen('inputPath/input.txt', 'w');
        $input = "input\\" . "\r\n" . $request->courseid . "\r\n" .
$request->quizid;
        fwrite($inputfile, $input);
        fclose($inputfile);
        exec("java -jar " . trim(env('PLAGIARISM'),""));
        chdir('..');
        return "true";
}
```

Kode Sumber 4.18 Mengeksekusi *File Executable* dari Modul *Plagiarism Detection System*

```
public function index($courseid, $quizid, $filter) {
	$course = Course::find($courseid);
	$quiz = Quiz::find($quizid);
	$enrollid = $quiz->enroll_id;

	$result = file_get_contents('plagiarism_plugin/outputCluster/Hasil_Cluster'.$courseid.$quizid.'.txt');

	if($result == "") {
		return redirect('enroll/'.$enrollid.'/quiz/'.$quizid.'/answer')->with('error', 'Pastikan mahasiswa telah mengumpulkan tugasnya');
	}

	$clusterdirty = explode('-------------------------------------------JUMLAH CLUSTER = ', $result);
	$i = 0;
	$listcluster = array();
	$fixedcluster = array();
	foreach ($clusterdirty as $scrumbled) {
		++$i;
		if($i == 1) {
			continue;
		}
		$numbersplited = explode('----------------------------------------', $scrumbled);
		$cluster = $numbersplited[0];
		$listcluster[] = $cluster;
		$fixedcluster[$cluster] = array();

		$memberdirty = $numbersplited[1];
		$stdsplited = preg_split('\"\r\n\r\n\"', $memberdirty);

		$clustermember = $stdsplited[0];
		$arrmember = preg_split('\"\r\n\"', $clustermember);
		$j = 0;
		foreach ($arrmember as $memberdirty) {
			++$j;
```

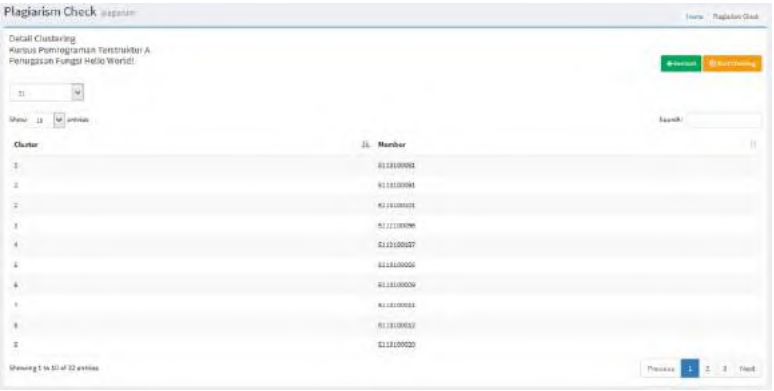
```
                        if($j == 1) {
                                continue;
                        }
                        $member = explode(' ', $memberdirty);
                        unset($member[count($member)-1]);
                        $fixedcluster[$cluster][] = $member;
                }

                $stdwithvar = $stdsplited[1];
                $std = trim($stdwithvar, "STD =");
                $fixedcluster[$cluster][] = $std;
        }

        if($filter == 0) {
                $filter = $listcluster[0];
        }

        return view('plagiarism.index', ['listcluster' => $listcluster,
'fixedcluster' => $fixedcluster[$filter], 'courseid' => $courseid,
'enrollid' => $enrollid, 'quizid' => $quizid, 'selected' => $filter, 'course'
=> $course, 'quiz' => $quiz]);
}
```

Kode Sumber 4.19 Mengambil Data dan Menampilkan *Cluster* Mahasiswa yang Memiliki Tingkat Kemiripan

Setelah dilakukan proses eksekusi, maka selanjutnya adalah mengambil data hasil eksekusi yang tersimpan di dalam *file* dan menampilkannya ke halaman *web*. Fungsi index seperti yang ditunjukkan pada Kode Sumber 4.19, bertugas untuk melakukan proses ini.

Antarmuka dari *Plagiarism Detection System* akan dipasang pada *role* Dosen. Sehingga pengguna dengan *role* Dosen bisa dengan mudah melakukan pengecekan pekerjaan mahasiswa. Pada Gambar 4.53 ditunjukkan bagaimana hasil proses pengecekan yang dilakukan, dengan kolom yang menunjukkan *cluster-cluster* dari mahasiswa-mahasiswa yang memiliki tingkat

kemiripan pekerjaan. *Cluster* yang ada bernilai 1 sampai dengan jumlah penugasan dari mahasiswa yang ada.



Gambar 4.53 Antarmuka Modul *Plagiarism Detection System*

Untuk penjelasan lebih jauh mengenai proses pengecekan *similarity* ini, pembaca bisa merujuk ke tugas akhir yang berjudul Rancang Bangun Sistem E-learning Pemrograman pada Modul Deteksi Plagiarisme Kode Program Mahasiswa dalam Satu Kelas yang disusun oleh saudari Ruchi Intan Tantra NRP 5112100015 [11].

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas mengenai Pengujian dari segi fungsionalitas aplikasi. Pengujian fungsionalitas dibagi menjadi beberapa skenario fungsionalitas yang terdapat pada aplikasi.

5.1. Lingkungan Pengujian

Dalam proses pengujian aplikasi dibutuhkan lingkungan pengujian yang disesuaikan standar kebutuhan. Pengujian aplikasi ini dilakukan dengan menggunakan satu laptop. Adapun spesifikasi laptop yang digunakan adalah sebagai berikut:

Perangkat	Laptop Toshiba Satellite L840
Sistem Operasi	Windows Embedded 8.1 Industry Pro 64-bit (6.3, Build 9600)
Prosesor	Intel(R) Core(TM) i7-3517U CPU @ 1.90 GHz (4 CPUs), ~2.4 GHz
RAM	4096 MB
Peramban	Mozilla Firefox 45.0.1

5.2. Dasar Pengujian

Pengujian yang dilakukan berupa pengujian fungsionalitas. Pengujian fungsionalitas dilakukan dengan model *black box* untuk masing-masing fungsionalitas dari aplikasi ini. Pengujian ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan diimplementasikan dengan baik dan bekerja seperti yang diharapkan.

5.3. Pengujian Fungsionalitas

Pada subbab ini, dijelaskan mengenai pengujian fungsionalitas dari sistem ini. Pengujian yang dilakukan direpresentasikan pada Tabel 5.1.

Tabel 5.1 Daftar Pengujian Fungsionalitas Sistem

Kode Pengujian	Uji Coba	Status
TC-FR-01	Melihat Daftar Konvensi Gaya Penulisan Kode	Berhasil
TC-FR-02	Membuat Konvensi Gaya Penulisan Kode Baru	Berhasil
TC-FR-03	Melihat Detail Konvensi Gaya Penulisan Kode	Berhasil
TC-FR-04	Mengedit Konvensi Gaya Penulisan Kode	Berhasil
TC-FR-05	Menghapus Konvensi Gaya Penulisan Kode	Berhasil
TC-FR-06	Melihat Daftar Kursus	Berhasil
TC-FR-07	Membuat Kursus Baru	Berhasil
TC-FR-08	Melihat Detail Kursus	Berhasil
TC-FR-09	Mengedit Kursus	Berhasil
TC-FR-10	Menghapus Kursus	Berhasil
TC-FR-11	Mengenroll User ke dalam Kursus	Berhasil
TC-FR-12	Melihat Timeline Pengumuman	Berhasil
TC-FR-13	Melihat Kursus yang Diikuti	Berhasil
TC-FR-14	Melihat Daftar Penugasan	Berhasil
TC-FR-15	Membuat Penugasan Baru	Berhasil
TC-FR-16	Melihat Detail Penugasan	Berhasil
TC-FR-17	Mengedit Penugasan	Berhasil
TC-FR-18	Menghapus Penugasan	Berhasil
TC-FR-19	Melihat Daftar Jawaban	Berhasil
TC-FR-20	Membuat Jawaban Baru	Berhasil
TC-FR-21	Melihat Detail Jawaban	Berhasil
TC-FR-22	Mengedit Jawaban	Berhasil

TC-FR-23	Melakukan Pengecekan Kode Program Secara Instan	Berhasil
TC-FR-24	Melakukan Percobaan Compile Kode Program	Berhasil

5.3.1. Pengujian Melihat Daftar Konvensi Gaya Penulisan Kode

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Konvensi Gaya Penulisan Kode terkait tentang menampilkan daftar konvensi gaya penulisan kode. Skenario pengujian dijelaskan pada Tabel 5.2.

Tabel 5.2 Pengujian Melihat Daftar Konvensi Gaya Penulisan Kode

Kode Pengujian	TC-FR-01
Referensi Kasus Penggunaan	UC-01
Nama	Pengujian Melihat Daftar Konvensi Gaya Penulisan Kode
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan daftar konvensi gaya penulisan kode yang ada
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Konvensi Gaya Penulisan Kode pada *navbar*
Hasil Yang Diharapkan	Sistem menampilkan daftar konvensi gaya penulisan kode yang ada
Hasil Yang Didapat	Sistem menampilkan daftar konvensi gaya penulisan kode yang ada
Hasil Pengujian	Berhasil

5.3.2. Membuat Konvensi Gaya Penulisan Kode Baru

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Konvensi Gaya Penulisan Kode terkait tentang pembuatan konvensi gaya penulisan kode baru. Skenario pengujian dijelaskan pada Tabel 5.3.

Tabel 5.3 Pengujian Membuat Konvensi Gaya Penulisan Kode Baru

Kode Pengujian	TC-FR-02
Referensi Kasus Penggunaan	UC-02
Nama	Pengujian Membuat Konvensi Gaya Penulisan Kode Baru
Tujuan Pengujian	Menguji apakah sistem dapat menyimpan konvensi gaya penulisan kode baru ke dalam basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Konvensi Gaya Penulisan Kode pada *navbar* - Pengguna memilih tombol Tambah - Pengguna mengisi isian dengan tepat - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan konvensi kode baru berhasil ditambahkan
Hasil Yang Didapat	Sistem menampilkan pesan konvensi kode baru berhasil ditambahkan
Hasil Pengujian	Berhasil

5.3.3. Melihat Detail Konvensi Gaya Penulisan Kode

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Konvensi Gaya Penulisan Kode terkait tentang menampilkan detail konvensi gaya penulisan kode. Skenario pengujian dijelaskan pada Tabel 5.4.

Tabel 5.4 Pengujian Melihat Detail Konvensi Gaya Penulisan Kode

Kode Pengujian	TC-FR-03
Referensi Kasus Penggunaan	UC-03
Nama	Pengujian Melihat Detail Konvensi Gaya Penulisan Kode
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan detail dari suatu konvensi gaya penulisan kode
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Konvensi Gaya Penulisan Kode pada *navbar* - Pengguna memilih tombol Detail pada kolom Aksi dari salah satu konvensi gaya penulisan kode yang ada
Hasil Yang Diharapkan	Sistem dapat menampilkan detail dari suatu konvensi gaya penulisan kode
Hasil Yang Didapat	Sistem dapat menampilkan detail dari suatu konvensi gaya penulisan kode
Hasil Pengujian	Berhasil

5.3.4. Mengedit Konvensi Gaya Penulisan Kode

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Konvensi Gaya Penulisan Kode terkait tentang pengeditan konvensi gaya penulisan kode yang ada. Skenario pengujian dijelaskan pada Tabel 5.5.

Tabel 5.5 Pengujian Mengedit Konvensi Gaya Penulisan Kode

Kode Pengujian	TC-FR-04
Referensi Kasus Penggunaan	UC-04
Nama	Pengujian Mengedit Konvensi Gaya Penulisan Kode
Tujuan Pengujian	Menguji apakah sistem dapat memperbarui konvensi gaya penulisan kode yang telah ada di dalam basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Konvensi Gaya Penulisan Kode pada *navbar* - Pengguna memilih tombol Edit pada kolom Aksi dari salah satu konvensi gaya penulisan kode yang ada - Pengguna mengedit isian dengan tepat - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan konvensi kode berhasil diupdate
Hasil Yang Didapat	Sistem menampilkan pesan konvensi kode berhasil diupdate
Hasil Pengujian	Berhasil

5.3.5. Menghapus Konvensi Gaya Penulisan Kode

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Konvensi Gaya Penulisan Kode terkait tentang penghapusan konvensi gaya penulisan kode. Skenario pengujian dijelaskan pada Tabel 5.6.

Tabel 5.6 Pengujian Menghapus Konvensi Gaya Penulisan Kode

Kode Pengujian	TC-FR-05
Referensi Kasus Penggunaan	UC-05
Nama	Pengujian Menghapus Konvensi Gaya Penulisan Kode
Tujuan Pengujian	Menguji apakah sistem dapat menghapus suatu konvensi gaya penulisan kode dari basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Konvensi Gaya Penulisan Kode pada *navbar* - Pengguna memilih tombol Hapus pada kolom Aksi dari salah satu konvensi gaya penulisan kode yang ada - Pengguna memilih tombol Yakin pada dialog konfirmasi penghapusan konvensi gaya penulisan kode
Hasil Yang Diharapkan	Sistem menampilkan pesan konvensi kode berhasil dihapus
Hasil Yang Didapat	Sistem menampilkan pesan konvensi kode berhasil dihapus
Hasil Pengujian	Berhasil

5.3.6. Melihat Daftar Kursus

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Kursus terkait tentang menampilkan daftar kursus. Skenario pengujian dijelaskan pada Tabel 5.7.

Tabel 5.7 Pengujian Melihat Daftar Kursus

Kode Pengujian	TC-FR-06
Referensi Kasus Penggunaan	UC-06
Nama	Pengujian Melihat Daftar Kursus
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan daftar kursus yang ada
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Kursus pada *navbar*
Hasil Yang Diharapkan	Sistem menampilkan daftar kursus yang ada
Hasil Yang Didapat	Sistem menampilkan daftar kursus yang ada
Hasil Pengujian	Berhasil

5.3.7. Membuat Kursus Baru

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Kursus terkait tentang pembuatan kursus baru. Skenario pengujian dijelaskan pada Tabel 5.8.

Tabel 5.8 Pengujian Membuat Kursus Baru

Kode Pengujian	TC-FR-07
Referensi Kasus Penggunaan	UC-07
Nama	Pengujian Membuat Kursus Baru
Tujuan Pengujian	Menguji apakah sistem dapat menyimpan kursus baru ke dalam basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Kursus pada *navbar* - Pengguna memilih tombol Tambah - Pengguna mengisi isian dengan tepat - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan kursus baru berhasil ditambahkan
Hasil Yang Didapat	Sistem menampilkan pesan kursus baru berhasil ditambahkan
Hasil Pengujian	Berhasil

5.3.8. Melihat Detail Kursus

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Kursus terkait tentang menampilkan detail kursus. Skenario pengujian dijelaskan pada Tabel 5.9.

Tabel 5.9 Pengujian Melihat Detail Kursus

Kode Pengujian	TC-FR-08
Referensi Kasus Penggunaan	UC-08
Nama	Pengujian Melihat Detail Kursus
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan detail dari suatu kursus
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Kursus pada *navbar* - Pengguna memilih tombol Detail pada kolom Aksi dari salah satu kursus yang ada
Hasil Yang Diharapkan	Sistem dapat menampilkan detail dari suatu kursus
Hasil Yang Didapat	Sistem dapat menampilkan detail dari suatu kursus
Hasil Pengujian	Berhasil

5.3.9. Mengedit Kursus

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Kursus terkait tentang pengeditan kursus yang ada. Skenario pengujian dijelaskan pada Tabel 5.10.

Tabel 5.10 Pengujian Mengedit Kursus

Kode Pengujian	TC-FR-09
Referensi Kasus Penggunaan	UC-09
Nama	Pengujian Mengedit Kursus
Tujuan Pengujian	Menguji apakah sistem dapat memperbarui kursus yang telah ada di dalam basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Kursus pada *navbar* - Pengguna memilih tombol Edit pada kolom Aksi dari salah satu kursus yang ada - Pengguna mengedit isian dengan tepat - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan kursus berhasil diupdate
Hasil Yang Didapat	Sistem menampilkan pesan kursus berhasil diupdate
Hasil Pengujian	Berhasil

5.3.10. Menghapus Kursus

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Kursus terkait tentang penghapusan kursus. Skenario pengujian dijelaskan pada Tabel 5.11.

Tabel 5.11 Pengujian Menghapus Kursus

Kode Pengujian	TC-FR-10
Referensi Kasus Penggunaan	UC-10
Nama	Pengujian Menghapus Kursus
Tujuan Pengujian	Menguji apakah sistem dapat menghapus suatu kursus dari basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar - Kursus yang akan dihapus tidak direferensikan oleh data lain
Langkah Pengujian	- Pengguna memilih menu Kursus pada *navbar* - Pengguna memilih tombol Hapus pada kolom Aksi dari salah satu kursus yang ada - Pengguna memilih tombol Yakin pada dialog konfirmasi penghapusan kursus
Hasil Yang Diharapkan	Sistem menampilkan pesan kursus berhasil dihapus
Hasil Yang Didapat	Sistem menampilkan pesan kursus berhasil dihapus
Hasil Pengujian	Berhasil

5.3.11. Mengenroll User ke dalam Kursus

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Kursus terkait tentang peng-*enroll*-an *user* ke dalam kursus. Skenario pengujian dijelaskan pada Tabel 5.12.

Tabel 5.12 Pengujian Mengenroll User ke dalam Kursus

Kode Pengujian	TC-FR-11
Referensi Kasus Penggunaan	UC-11
Nama	Pengujian Mengenroll User ke dalam Kursus
Tujuan Pengujian	Menguji apakah sistem dapat melakukan sinkronisasi data *user* dengan data kursus
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Kursus pada *navbar* - Pengguna memilih tombol Enrollment pada kolom Aksi dari salah satu kursus yang ada - Pengguna memilih tombol Edit - Pengguna melakukan *check* atau *uncheck* pada daftar *user* - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan enrollment berhasil diupdate
Hasil Yang Didapat	Sistem menampilkan pesan enrollment berhasil diupdate
Hasil Pengujian	Berhasil

5.3.12. Melihat Timeline Pengumuman

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu awal pada saat pengguna berhasil *login* terkait tentang menampilkan *timeline* pengumuman. Skenario pengujian dijelaskan pada Tabel 5.13.

Tabel 5.13 Pengujian Melihat Timeline Pengumuman

Kode Pengujian	TC-FR-12
Referensi Kasus Penggunaan	UC-12
Nama	Pengujian Melihat Timeline Pengumuman
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan pengumuman dalam bentuk *timeline*
Kondisi Awal	- Pengguna adalah *user* dengan *role* Administrator, Dosen, atau Mahasiswa - Pengguna belum melakukan *login*
Langkah Pengujian	- Pengguna melakukan *login* dengan benar
Hasil Yang Diharapkan	Sistem dapat menampilkan pengumuman dalam bentuk *timeline*
Hasil Yang Didapat	Sistem dapat menampilkan pengumuman dalam bentuk *timeline*
Hasil Pengujian	Berhasil

5.3.13. Melihat Kursus yang Diikuti

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem pada menu Enroll terkait tentang menampilkan daftar kursus yang telah di-*enroll*. Skenario pengujian dijelaskan pada Tabel 5.14.

Tabel 5.14 Pengujian Melihat Kursus yang Diikuti

Kode Pengujian	TC-FR-13
Referensi Kasus Penggunaan	UC-13

Nama	Pengujian Melihat Kursus yang Diikuti
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan daftar kursus yang diikuti oleh seorang *user*
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen atau Mahasiswa - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar*
Hasil Yang Diharapkan	Sistem menampilkan daftar kursus yang diikuti oleh *user*
Hasil Yang Didapat	Sistem menampilkan daftar kursus yang diikuti oleh *user*
Hasil Pengujian	Berhasil

5.3.14. Melihat Daftar Penugasan

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang menampilkan daftar penugasan dalam suatu kursus. Skenario pengujian dijelaskan pada Tabel 5.15.

Tabel 5.15 Pengujian Melihat Daftar Penugasan

Kode Pengujian	TC-FR-14
Referensi Kasus Penggunaan	UC-14
Nama	Pengujian Melihat Daftar Penugasan
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan daftar penugasan yang ada di dalam suatu kursus
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen atau Mahasiswa

	- Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar* - Pengguna memilih salah satu kursus yang ada
Hasil Yang Diharapkan	Sistem menampilkan daftar penugasan yang ada dalam suatu kursus
Hasil Yang Didapat	Sistem menampilkan daftar penugasan yang ada dalam suatu kursus
Hasil Pengujian	Berhasil

5.3.15. Membuat Penugasan Baru

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang pembuatan penugasan baru dalam suatu kursus. Skenario pengujian dijelaskan pada Tabel 5.16.

Tabel 5.16 Pengujian Membuat Penugasan Baru

Kode Pengujian	TC-FR-15
Referensi Kasus Penggunaan	UC-15
Nama	Pengujian Membuat Penugasan Baru
Tujuan Pengujian	Menguji apakah sistem dapat menyimpan penugasan baru ke dalam basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar*

	- Pengguna memilih salah satu kursus yang ada - Pengguna memilih tombol Tambah - Pengguna mengisi isian dengan tepat - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan penugasan baru berhasil ditambahkan
Hasil Yang Didapat	Sistem menampilkan pesan penugasan baru berhasil ditambahkan
Hasil Pengujian	Berhasil

5.3.16. Melihat Detail Penugasan

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang menampilkan detail penugasan dalam suatu kursus. Skenario pengujian dijelaskan pada Tabel 5.17.

Tabel 5.17 Pengujian Melihat Detail Penugasan

Kode Pengujian	TC-FR-16
Referensi Kasus Penggunaan	UC-16
Nama	Pengujian Melihat Detail Penugasan
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan detail dari suatu penugasan
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen atau Mahasiswa - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar*

	- Pengguna memilih salah satu kursus yang ada - Pengguna memilih tombol Detail pada kolom Aksi dari salah satu penugasan yang ada
Hasil Yang Diharapkan	Sistem dapat menampilkan detail dari suatu penugasan
Hasil Yang Didapat	Sistem dapat menampilkan detail dari suatu penugasan
Hasil Pengujian	Berhasil

5.3.17. Mengedit Penugasan

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang pengeditan penugasan dalam suatu kursus. Skenario pengujian dijelaskan pada Tabel 5.18.

Tabel 5.18 Pengujian Mengedit Penugasan

Kode Pengujian	TC-FR-17
Referensi Kasus Penggunaan	UC-17
Nama	Pengujian Mengedit Penugasan
Tujuan Pengujian	Menguji apakah sistem dapat memperbarui penugasan yang telah ada di dalam basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen - Pengguna telah melakukan *login* dengan benar
Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar* - Pengguna memilih salah satu kursus yang ada

	- Pengguna memilih tombol Edit pada kolom Aksi dari salah satu penugasan yang ada - Pengguna mengisi isian dengan tepat - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan penugasan berhasil diupdate
Hasil Yang Didapat	Sistem menampilkan pesan penugasan berhasil diupdate
Hasil Pengujian	Berhasil

5.3.18. Menghapus Penugasan

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang penghapusan penugasan dalam suatu kursus. Skenario pengujian dijelaskan pada Tabel 5.19.

Tabel 5.19 Pengujian Menghapus Penugasan

Kode Pengujian	TC-FR-18
Referensi Kasus Penggunaan	UC-18
Nama	Pengujian Menghapus Penugasan
Tujuan Pengujian	Menguji apakah sistem dapat menghapus suatu penugasan dari basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen - Pengguna telah melakukan *login* dengan benar - Penugasan yang akan dihapus tidak direferensikan oleh data lain
Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar*

	- Pengguna memilih salah satu kursus yang ada - Pengguna memilih tombol Hapus pada kolom Aksi dari salah satu penugasan yang ada - Pengguna memilih tombol Yakin pada dialog konfirmasi penghapusan penugasan
Hasil Yang Diharapkan	Sistem menampilkan pesan penugasan berhasil dihapus
Hasil Yang Didapat	Sistem menampilkan pesan penugasan berhasil dihapus
Hasil Pengujian	Berhasil

5.3.19. Melihat Daftar Jawaban

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang menampilkan daftar jawaban dalam suatu penugasan. Skenario pengujian dijelaskan pada Tabel 5.20.

Tabel 5.20 Pengujian Melihat Daftar Jawaban

Kode Pengujian	TC-FR-19
Referensi Kasus Penggunaan	UC-19
Nama	Pengujian Melihat Daftar Jawaban
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan daftar jawaban yang ada di dalam suatu penugasan
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen atau Mahasiswa - Pengguna telah melakukan *login* dengan benar

Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar* - Pengguna memilih salah satu kursus yang ada - Pengguna memilih tombol Jawaban pada kolom Aksi dari salah satu penugasan yang ada
Hasil Yang Diharapkan	Sistem menampilkan daftar jawaban yang ada dalam suatu penugasan
Hasil Yang Didapat	Sistem menampilkan daftar jawaban yang ada dalam suatu penugasan
Hasil Pengujian	Berhasil

5.3.20. Membuat Jawaban Baru

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang pembuatan jawaban baru dalam suatu penugasan. Skenario pengujian dijelaskan pada Tabel 5.21.

Tabel 5.21 Pengujian Membuat Jawaban Baru

Kode Pengujian	TC-FR-20
Referensi Kasus Penggunaan	UC-20
Nama	Pengujian Membuat Jawaban Baru
Tujuan Pengujian	Menguji apakah sistem dapat menyimpan jawaban baru ke dalam basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Mahasiswa - Pengguna telah melakukan *login* dengan benar

Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar* - Pengguna memilih salah satu kursus yang ada - Pengguna memilih tombol Jawaban pada kolom Aksi dari salah satu penugasan yang ada - Pengguna memilih tombol Tambah Jawaban pada kolom Status Pengerjaan - Pengguna melakukan proses penulisan kode program - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan jawaban baru berhasil ditambahkan
Hasil Yang Didapat	Sistem menampilkan pesan jawaban baru berhasil ditambahkan
Hasil Pengujian	Berhasil

5.3.21. Melihat Detail Jawaban

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang menampilkan detail jawaban dalam suatu penugasan. Skenario pengujian dijelaskan pada Tabel 5.22.

Tabel 5.22 Pengujian Melihat Detail Jawaban

Kode Pengujian	TC-FR-21
Referensi Kasus Penggunaan	UC-21
Nama	Pengujian Melihat Detail Jawaban
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan detail dari suatu jawaban

Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen atau Mahasiswa - Pengguna telah melakukan *login* dengan benar - Jawaban yang akan dilihat detailnya sudah pernah dikerjakan sebelumnya
Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar* - Pengguna memilih salah satu kursus yang ada - Pengguna memilih tombol Jawaban pada kolom Aksi dari salah satu penugasan yang ada - Pengguna memilih tombol Lihat Jawaban pada kolom Status Pengerjaan dari salah satu peserta yang ada
Hasil Yang Diharapkan	Sistem dapat menampilkan detail dari suatu jawaban
Hasil Yang Didapat	Sistem dapat menampilkan detail dari suatu jawaban
Hasil Pengujian	Berhasil

5.3.22. Mengedit Jawaban

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem terkait tentang pengeditan jawaban dalam suatu penugasan. Skenario pengujian dijelaskan pada Tabel 5.23.

Tabel 5.23 Pengujian Mengedit Jawaban

Kode Pengujian	TC-FR-22
Referensi Kasus Penggunaan	UC-22

Nama	Pengujian Mengedit Jawaban
Tujuan Pengujian	Menguji apakah sistem dapat memperbarui jawaban yang telah ada di dalam basis data
Kondisi Awal	- Pengguna adalah *user* dengan *role* Mahasiswa - Pengguna telah melakukan *login* dengan benar - Jawaban yang akan diedit sudah pernah dikerjakan sebelumnya
Langkah Pengujian	- Pengguna memilih menu Enroll pada *navbar* - Pengguna memilih salah satu kursus yang ada - Pengguna memilih tombol Jawaban pada kolom Aksi dari salah satu penugasan yang ada - Pengguna memilih tombol Edit Jawaban pada kolom Status Pengerjaan - Pengguna melakukan proses penulisan kode program - Pengguna memilih tombol Simpan
Hasil Yang Diharapkan	Sistem menampilkan pesan jawaban berhasil diupdate
Hasil Yang Didapat	Sistem menampilkan pesan jawaban berhasil diupdate
Hasil Pengujian	Berhasil

5.3.23. Melakukan Pengecekan Kode Program Secara Instan

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem untuk melakukan pengecekan pada kode program yang ditulis oleh Dosen atau Mahasiswa secara instan mengenai masalah *syntax* dan *code styling convention*. Skenario pengujian dijelaskan pada Tabel 5.24.

Tabel 5.24 Pengujian Melakukan Pengecekan Kode Program Secara Instan

Kode Pengujian	TC-FR-23
Referensi Kasus Penggunaan	UC-23
Nama	Pengujian Melakukan Pengecekan Kode Program Secara Instan
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan umpan balik atas pengecekan *syntax* dan *code styling convention* secara instan
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen atau Mahasiswa - Pengguna telah melakukan *login* dengan benar - Pengguna bisa mengakses Ace Editor yang ada pada sistem
Langkah Pengujian	- Pengguna melakukan penulisan kode program pada Ace Editor
Hasil Yang Diharapkan	Sistem dapat secara langsung mengoreksi kode program yang ditulis, dan jika terjadi *syntax error* atau *code convention error*, maka pengguna akan langsung diberikan peringatan lewat Ace Editor berupa tanda *error* dan *warning*
Hasil Yang Didapat	Sistem dapat secara langsung mengoreksi kode program yang ditulis, dan jika terjadi *syntax error* atau *code convention error*, maka pengguna akan langsung diberikan peringatan lewat Ace Editor berupa tanda *error* dan *warning*
Hasil Pengujian	Berhasil

5.3.24. Melakukan Percobaan Compile Kode Program

Skenario pengujian ini dibuat untuk mengetahui fungsionalitas sistem untuk melakukan percobaan kompilasi kode program yang telah ditulis oleh Dosen atau Mahasiswa. Skenario pengujian dijelaskan pada Tabel 5.25.

Tabel 5.25 Pengujian Melakukan Percobaan Compile Kode Program

Kode Pengujian	TC-FR-24
Referensi Kasus Penggunaan	UC-24
Nama	Pengujian Melakukan Percobaan Compile Kode Program
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan umpan balik atas percobaan kompilasi kode program menggunakan *compiler* lokal
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen atau Mahasiswa - Pengguna telah melakukan *login* dengan benar - Pengguna bisa mengakses Ace Editor yang ada pada sistem
Langkah Pengujian	- Pengguna melakukan penulisan kode program pada Ace Editor - Pengguna memilih tombol Compilability Test
Hasil Yang Diharapkan	Sistem memberikan umpan balik berupa pemberitahuan apakah kode program *compileable* atau tidak

Hasil Yang Didapat	Sistem memberikan umpan balik berupa pemberitahuan apakah kode program *compileable* atau tidak
Hasil Pengujian	Berhasil

5.4. Pengujian Non-Fungsional

Pada subbab ini dijelaskan pengujian terhadap kebutuhan non-fungsional pada sistem yaitu berkaitan dengan hak akses. Pengujian yang dilakukan, direpresentasikan pada Tabel 5.26.

Tabel 5.26 Daftar Pengujian Hak Akses Pengguna

Hak Akses	Menu yang Bisa Diakses	Status
Admin	Manajemen Konvensi Gaya Penulisan Kode	Berhasil
	Manajemen Kursus	Berhasil
Dosen dan Mahasiswa	Daftar Kursus yang Di-*enroll*	Berhasil
Dosen	Manajemen Penugasan (hak akses penuh)	Berhasil
	Manajemen Jawaban (hak akses sebagian)	Berhasil
Mahasiswa	Manajemen Penugasan (hak akses sebagian)	Berhasil
	Manajemen Jawaban (hak akses penuh)	Berhasil

5.5. Pengujian *Usability*

Usability berasal dari kat *usable* yang secara umum berarti dapat digunakan dengan baik. Sesuatu dapat dikatakan berguna dengan baik apabila kegagalan dalam penggunaannya dapat dihilangkan atau dimiminalkan, serta memberi manfaat dan kepuasan pada pengguna.

Usability mengacu pada bagaimana pengguna bisa mempelajari dan menggunakan produk untuk memperoleh tujuannya, dan seberapa puas mereka terhadap penggunaannya.

Definisi *usability* menurut ISO 9241:11 adalah sejauh mana suatu produk dapat digunakan oleh pengguna tertentu untuk mencapai target yang ditetapkan dengan efektivitas, efisiensi, dan mencapai kepuasan penggunaan dalam konteks tertentu. Konteks penggunaan terdiri dari pengguna, tugas, dan peralatan (*hardware, software*, dan material). Berdasarkan definisi tersebut *usability* diukur berdasarkan beberapa komponen, yaitu *learnability*, *efficiency*, *memorability*, *error*, dan *satisfaction*.

5.5.1. Pengujian *Usability* dengan *User Questionnaire*

Kuesioner adalah instrumen penelitian yang berupa daftar pertanyaan untuk memperoleh keterangan dari sejumlah responden (sumber yang diambil datanya). Kuesioner dapat disebut sebagai wawancara tertulis, karena isi kuesioner merupakan satu rangkaian pertanyaan tertulis yang ditujukan kepada responden dan diisi sendiri oleh responden. Dalam pengujian *usability* ini, akan dikumpulkan data dengan menggunakan kuesioner.

Langkah-langkah yang dilakukan untuk membuat sebuah kuesioner adalah dengan menentukan responden, menetapkan kategori yang akan diteliti, menentukan *task-task testing*, dan membuat daftar pertanyaan untuk kuesioner. Setelah mendapatkan hasil pengujian dengan menggunakan kuesioner, analisis data pun dapat dilakukan.

5.5.2. Responden

Responden adalah orang yang menjawab pertanyaan yang diajukan pada sebuah kuesioner. Pada pengujian ini, responden berjumlah 5 orang. Pada Tabel 5.27 terdapat daftar responden untuk pengujian *usability* ini.

Tabel 5.27 Daftar Responden

No.	Nama Responden
1.	Ardhya Perdana Putra
2.	Fandy Ahmad
3.	Argyanto Dimas N.
4.	Muhammad Ardhiansyah Metana Putra
5.	Fananda Herda Perdana

5.5.3. Kategori Pengujian

Ada lima kategori yang diujikan pada proses pengujian ini. Kelima kategori tersebut antara lain adalah *learnability*, *efficiency*, *memorability*, *error*, dan *satisfaction*.

- Kemudahan (*learnability*), menjelaskan tingkat kemudahan pengguna dalam mempelajari sistem untuk memenuhi tugas-tugas dasar ketika pertama kali menggunakan sistem tersebut.
- Efisiensi (*efficiency*), menjelaskan tingkat kecepatan pengguna dalam menyelesaikan tugas-tugas setelah mempelajari program.
- Mudah diingat (*memorability*), menjelaskan tingkat kemudahan pengguna dalam menggunakan program dengan baik, setelah lama tidak menggunakan.
- Kesalahan (*error*), menjelaskan kesalahan yang terjadi karena kesalahan pengguna atau sistem.
- Kepuasan (*satisfaction*), menjelaskan tingkat kepuasan pengguna dalam menggunakan program.

5.5.4. *Task-task Usability Testing*

Langkah awal *usability* ini adalah memberikan sejumlah *task* atau tugas yang sudah dipersiapkan sebelumnya kepada pengguna saat berinteraksi dengan sistem yang diuji. *Task-task* ini diberikan kepada 5 responden yang telah dipilih. *Task-task* ini digunakan sebagai sarana interaksi dalam pengukuran *usability*. *Task-task* ini terdapat pada Tabel 5.28.

Tabel 5.28 *Task-task Usability Testing*

No.	***Task* / Tugas**
1.	Melakukan *login*
2.	Menampilkan daftar konvensi gaya penulisan kode
3.	Menambah konvensi gaya penulisan kode baru
4.	Menampilkan detail konvensi gaya penulisan kode
5.	Mengubah konvensi gaya penulisan kode
6.	Menghapus konvensi gaya penulisan kode
7.	Melihat daftar kursus
8.	Menambah kursus baru
9.	Melihat detail kursus
10.	Mengubah kursus
11.	Menghapus kursus
12.	Memasukkan mahasiswa ke dalam kursus
13.	Melihat kursus yang diikuti
14.	Melihat daftar penugasan
15.	Menambah penugasan baru
16.	Melihat detail penugasan
17.	Mengedit penugasan
18.	Menghapus penugasan
19.	Melihat daftar jawaban
20.	Menambah jawaban baru
21.	Melihat detail jawaban dan riwayat pengerjaan
22.	Mengedit jawaban hingga sesuai dengan konvensi gaya penulisan kode

5.5.5. Daftar Pertanyaan

Berdasarkan ketegori pengujian dan *task-task* yang telah ditentukan, maka dibuat daftar pertanyaan yang harus diisi oleh responden. Daftar pertanyaan ini dapat dilihat pada Tabel 5.29.

Tabel 5.29 Daftar Pertanyaan Kuesioner

No.	Pertanyaan
Kemudahan (*learnability*)	
1.	Tulisan teks yang digunakan sudah jelas
2.	Bahasa yang digunakan sistem sudah jelas
3.	Menu-menu yang disediakan sistem cukup mudah dipahami
Efisiensi (*efficiency*)	
4.	Saat menu diklik, tampilan selanjutnya akan berganti dengan cepat
5.	Proses *login* dan *logout* berlangsung dengan cepat
6.	Tidak pernah terjadi *loading* yang lama
Mudah diingat (*memorability*)	
7.	Nama halaman sistem ini adalah “ELearning”
8.	Tampilan sistem didominasi oleh warna biru
9.	Sistem menampilkan daftar pengumuman setelah anda berhasil *login*
Kesalahan (*error*)	
10.	Anda tidak menemukan *link* yang anda klik *error*
11.	Sistem memberikan umpan balik saat pengguna melakukan *coding*
12.	Diagram riwayat pengerjaan yang dilakukan oleh mahasiswa muncul
Kepuasan (*satisfaction*)	
13.	Anda ingin menggunakan sistem ini kembali
14.	Sistem ini memudahkan anda mendapatkan informasi yang anda inginkan
15.	Sistem ini membantu anda menyelesaikan pekerjaan anda

5.5.6. Skala Pengukuran

Skala pengukuran terhadap suatu obyek terdiri dari empat macam, yaitu skala nominal, skala ordinal, skala interval, dan skala rasio. Keempat skala pengukuran ini jika digunakan di dalam kuesioner dapat dilakukan dengan salah satu instrument pendekatan skala *likert*. Skala *likert* digunakan untuk mengukur tanggapan atau respon yang berhubungan dengan pernyataan tentang sikap seseorang. Alternatif pernyataan misalnya dari sangat setuju sampai sangat tidak setuju, setiap item diberi pilihan respon yang sifatnya tertutup. Banyaknya pilihan respon terdiri dari 3, 5, 7, 9, 11. Namun di dalam penerapan evaluasi *usability* digunakan skala 4 pilihan respon seperti pada Tabel 5.30. Tingkat pengukuran data dalam skala *likert* adalah ordinal sehingga apabila akan dianalisis dengan statistik harus dinaikkan terlebih dahulu menjadi skala interval.

Tabel 5.30 Skala Pengukuran

Skala	Keterangan
4	Sangat setuju
3	Setuju
2	Tidak setuju
1	Sangat tidak setuju

5.5.7. Analisis Data Hasil *Usability Test*

Menghitung persentase hasil *usability test* dengan menggunakan metode kuesioner dapat dilakukan dengan menggunakan rumus pada Persamaan (5.1).

$$HA = \frac{\text{n}}{4 \text{ x r}} \; x \; 100\% \qquad \textbf{(5.1)}$$

dengan:

HA = jumlah hasil akhir persentase;
n = jumlah nilai kuesioner tiap pertanyaan; dan
r = jumlah responden.

Menurut Arikunto, tabel kuantitatif untuk hasil perhitungan terhadap kuesioner pada *usability testing* untuk mengukur penggunaan sistem ini adalah seperti pada Tabel 5.31.

Tabel 5.31 Skala Pengukuran Penggunaan Sistem

Range	Kualifikasi	Hasil
85 – 100%	Sangat baik	Berhasil
65 – 84%	Baik	Berhasil
55 – 64%	Cukup	Tidak berhasil
0 – 54%	Kurang	Tidak berhasil

Pada Tabel 5.32 ditunjukkan data hasil survei yang dilakukan.

Tabel 5.32 Data Hasil Survei

No.	**Pertanyaan**	**Jawaban**					**n**
		R1	**R2**	**R3**	**R4**	**R5**	
Kemudahan (*learnability*)							
A1.	Tulisan teks yang digunakan sudah jelas	3	3	3	3	4	16
A2.	Bahasa yang digunakan sistem sudah jelas	3	3	4	3	3	16
A3.	Menu-menu yang disediakan sistem cukup mudah dipahami	3	2	3	3	4	15
Efisiensi (*efficiency*)							
A4.	Saat menu diklik, tampilan selanjutnya akan berganti dengan cepat	4	4	4	3	4	19
A5.	Proses *login* dan *logout* berlangsung dengan cepat	3	4	4	3	4	18

A6.	Tidak pernah terjadi *loading* yang lama	2	3	3	3	2	13
Mudah diingat (*memorability*)							
A7.	Nama halaman sistem ini adalah "ELearning"	3	4	4	3	4	18
A8.	Tampilan sistem didominasi oleh warna biru	2	3	2	3	4	14
A9.	Sistem menampilkan daftar pengumuman setelah anda berhasil *login*	3	3	4	3	4	17
Kesalahan (*error*)							
A10.	Anda tidak menemukan *link* yang anda klik *error*	3	4	4	3	4	18
A11.	Sistem memberikan umpan balik saat pengguna melakukan *coding*	3	3	4	3	4	17
A12.	Diagram riwayat pengerjaan yang dilakukan oleh mahasiswa muncul	3	3	4	3	4	17
Kepuasan (*satisfaction*)							
A13.	Anda ingin menggunakan sistem ini kembali	4	3	3	3	3	16
A14.	Sistem ini memudahkan anda mendapatkan informasi yang anda inginkan	3	3	3	3	4	16
A15.	Sistem ini membantu anda menyelesaikan pekerjaan anda	3	3	3	4	4	17

Dengan menggunakan rumus pada Persamaan (5.1), maka diperoleh hasil perhitungan seperti di bawah ini.

$$HA1 = \frac{16}{4 \text{ x } 5}\ x\ 100\% = \ 80\% \qquad HA9 = \frac{17}{4 \text{ x } 5}\ x\ 100\% = 85\%$$

$$HA2 = \frac{16}{4 \text{ x } 5}\ x\ 100\% = 80\% \qquad HA10 = \frac{18}{4 \text{ x } 5}\ x\ 100\% = 90\%$$

$$HA3 = \frac{15}{4 \text{ x } 5}\ x\ 100\% = 75\% \qquad HA11 = \frac{17}{4 \text{ x } 5}\ x\ 100\% = 85\%$$

$$HA4 = \frac{19}{4 \text{ x } 5}\ x\ 100\% = 95\% \qquad HA12 = \frac{17}{4 \text{ x } 5}\ x\ 100\% = 85\%$$

$$HA5 = \frac{18}{4 \text{ x } 5}\ x\ 100\% = 90\% \qquad HA13 = \frac{16}{4 \text{ x } 5}\ x\ 100\% = 80\%$$

$$HA6 = \frac{13}{4 \text{ x } 5}\ x\ 100\% = 65\% \qquad HA14 = \frac{16}{4 \text{ x } 5}\ x\ 100\% = 80\%$$

$$HA7 = \frac{18}{4 \text{ x } 5}\ x\ 100\% = 90\% \qquad HA15 = \frac{17}{4 \text{ x } 5}\ x\ 100\% = 85\%$$

$$HA8 = \frac{14}{4 \text{ x } 5}\ x\ 100\% = 70\%$$

5.5.8. Kesimpulan Hasil *Usability Test*

Untuk melakukan penarikan kesimpulan mengenai berapa persen kah sistem ini memiliki kegunaan dalam aspek *learnability*, *efficiency*, *memorability*, *error*, dan *satisfaction*, maka kategori akan dihitung rata-ratanya berdasarkan aspek penilaian pertanyaan dengan menggunakan rumus Persamaan (5.2).

$$PK = \frac{\text{HA1} + \text{HA2} + \cdots + \text{HAn}}{\text{AT}} \quad \textbf{(5.2)}$$

dengan:
PK = persentase kategori;
HA = jumlah hasil akhir persentase;
AT = jumlah pertanyaan.

Dengan menggunakan rumus pada Persamaan (5.2), maka diperoleh hasil perhitungan seperti di bawah ini.

$$PK1 = \frac{80\% + 80\% + 75\%}{3} = 78{,}33\%$$

$$PK2 = \frac{95\% + 90\% + 65\%}{3} = 83{,}33\%$$

$$PK3 = \frac{90\% + 70\% + 85\%}{3} = 81{,}67\%$$

$$PK4 = \frac{90\% + 85\% + 85\%}{3} = 86{,}67\%$$

$$PK5 = \frac{80\% + 80\% + 85\%}{3} = 81{,}67\%$$

Berdasarkan perhitungan di atas maka dapat diperoleh kesimpulan *usability* sistem seperti yang terdapat pada Tabel 5.33.

Tabel 5.33 Kesimpulan *Usability* Sistem

No.	Kategori *Usability*	Persentase	Kesimpulan	
			Kualifikasi	Hasil
1.	Kemudahan (*learnability*)	78,33%	Baik	Berhasil
2.	Efisiensi (*efficiency*)	83,33%	Baik	Berhasil

3.	Mudah diingat (*memorability*)	81,67%	Baik	Berhasil
4.	Kesalahan (*error*)	86,67%	Sangat baik	Berhasil
5.	Kepuasan (*satisfaction*)	81,67%	Baik	Berhasil

Berdasarkan pada Tabel 5.33, maka dapat dibaca lima hal penting yang didapatkan. Hal-hal tersebut sebagai berikut.

1. Sistem ini berhasil memiliki tingkat kemudahan penggunaan sebesar 78,33% dari *range* teratas 100% dengan kualifikasi nilai baik. Artinya, sistem ini mudah digunakan bagi para pengguna.
2. Sistem ini berhasil memiliki tingkat efisiensi penggunaan sebesar 83,33% dari *range* teratas 100% dengan kualifikasi nilai baik. Artinya, sistem ini efisien saat digunakan bagi para pengguna.
3. Sistem ini berhasil memiliki tingkat mudah diingat penggunaannya sebesar 81,67% dari *range* teratas 100% dengan kualifikasi nilai baik. Artinya, sistem ini mudah diingat oleh para pengguna.
4. Sistem ini berhasil memiliki tingkat kesalahan saat penggunaan sebesar 86,67% dari *range* teratas 100% dengan kualifikasi nilai sangat baik. Artinya, sistem ini tidak memiliki tingkat kesalahan yang besar yang ditemukan oleh para pengguna.
5. Sistem ini berhasil memiliki kepuasan penggunaan sebesar 81,67% dari *range* teratas 100% dengan kualifikasi nilai baik. Artinya, sistem ini memuaskan saat digunakan oleh para pengguna.

Untuk kuesioner dapat dilihat pada LAMPIRAN C – KUESIONER *USABILITY TEST*.

5.6. Pengujian Integrasi Modul *Student Feedback System*

Pengujian pada integrasi modul *Student Feedback System* ini terdiri dari dua bagian. Bagian yang pertama adalah menguji apakah *platform elearning* bisa menjalankan modul ini. Detail pengujian pada bagian ini dijelaskan pada Tabel 5.34.

Tabel 5.34 Pengujian Mengeksekusi Modul *Student Feedback System*

Nama	**Pengujian Mengeksekusi Modul *Student Feedback System***
Tujuan Pengujian	Menguji apakah sistem dapat menjalankan *plugin* dari modul *Student Feedback System*
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen - Pengguna telah melakukan *login* dengan benar - Pengguna masuk ke halaman daftar jawaban dari suatu penugasan
Langkah Pengujian	- Pengguna memilih tombol Student Feedback - Pengguna memilih tombol Start Checking
Hasil Yang Diharapkan	Sistem mengeksekusi *plugin* dari modul *Student Feedback System*
Hasil Yang Didapat	Sistem mengeksekusi *plugin* dari modul *Student Feedback System*
Hasil Pengujian	Berhasil

Bagian pengujian yang kedua adalah menguji apakah *platform elearning* bisa menampilkan data hasil eksekusi modul ini. Detail pengujian dapat dilihat pada Tabel 5.35.

Tabel 5.35 Pengujian Menampilkan Data Hasil Pengeksekusian Modul *Student Feedback System*

Nama	**Pengujian Menampilkan Data Hasil Pengeksekusian Modul *Student Feedback System***
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan data hasil pengeksekusian modul *Student Feedback System*
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen - Pengguna telah melakukan *login* dengan benar - Pengguna masuk ke halaman daftar jawaban dari suatu penugasan - Pengguna sebelumnya pernah melakukan eksekusi *plugin Student Feedback*
Langkah Pengujian	- Pengguna memilih tombol Student Feedback
Hasil Yang Diharapkan	Sistem menampilkan data hasil pengeksekusian modul *Student Feedback System*
Hasil Yang Didapat	Sistem menampilkan data hasil pengeksekusian modul *Student Feedback System*
Hasil Pengujian	Berhasil

5.7. Pengujian Integrasi Modul *Plagiarism Detection System*

Pengujian pada integrasi modul *Plagiarism Detection System* ini terdiri dari dua bagian. Bagian yang pertama adalah menguji apakah *platform elearning* bisa menjalankan modul ini. Detail pengujian pada bagian ini dijelaskan pada Tabel 5.36.

Tabel 5.36 Pengujian Mengeksekusi Modul *Plagiarism Detection System*

Nama	**Pengujian Mengeksekusi Modul *Plagiarism Detection System***
Tujuan Pengujian	Menguji apakah sistem dapat menjalankan *plugin* dari modul *Plagiarism Detection System*
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen - Pengguna telah melakukan *login* dengan benar - Pengguna masuk ke halaman daftar jawaban dari suatu penugasan
Langkah Pengujian	- Pengguna memilih tombol Plagiarism Check - Pengguna memilih tombol Start Checking
Hasil Yang Diharapkan	Sistem mengeksekusi *plugin* dari modul *Plagiarism Detection System*
Hasil Yang Didapat	Sistem mengeksekusi *plugin* dari modul *Plagiarism Detection System*
Hasil Pengujian	Berhasil

Bagian pengujian yang kedua adalah menguji apakah *platform elearning* bisa menampilkan data hasil eksekusi modul ini. Detail pengujian dapat dilihat pada Tabel 5.37.

Tabel 5.37 Pengujian Menampilkan Data Hasil Pengeksekusian Modul *Plagiarism Detection System*

Nama	**Pengujian Menampilkan Data Hasil Pengeksekusian Modul *Plagiarism Detection System***
Tujuan Pengujian	Menguji apakah sistem dapat menampilkan data hasil pengeksekusian modul *Plagiarism Detection System*
Kondisi Awal	- Pengguna adalah *user* dengan *role* Dosen - Pengguna telah melakukan *login* dengan benar - Pengguna masuk ke halaman daftar jawaban dari suatu penugasan - Pengguna sebelumnya pernah melakukan eksekusi *plugin Plagiarism Detection*
Langkah Pengujian	- Pengguna memilih tombol Plagiarism Check
Hasil Yang Diharapkan	Sistem menampilkan data hasil pengeksekusian modul *Plagiarism Detection System*
Hasil Yang Didapat	Sistem menampilkan data hasil pengeksekusian modul *Plagiarism Detection System*
Hasil Pengujian	Berhasil

5.8. Pengujian *Running Time* dari *Instant Feedback System*

Dalam pengujian ini, *testcase* yang digunakan adalah seperti pada Kode Sumber 5.1. Detail mengenai *running time* yang dibutuhkan oleh sistem dijelaskan pada Tabel 5.38. Data yang dijelaskan di sini adalah *running time* yang terjadi pada saat halaman baru saja *ready* (kondisi editor teks masih kosong) dan dimasukkan kode program secara *bulk*.

```
#include<iostream>
using namespace std;

void print() {
   int k;
   string abc;
   double variable_random;
   cout << "Hello World!" << endl;
}

int main() {
   int a = 2;
   print();
   return 0;
}
```

Kode Sumber 5.1 *Testcase* Pengujian *Running Time Instant Feedback System*

Tabel 5.38 Data *Running Time*

No.	Unsur Kode Program	*Running Time*
1.		502 ms
2.	#include<iostream>	503 ms
3.	#include<iostream> using namespace std;	1253 ms

4.	void print() {
	cout << "Hello World!" << endl;
	}
		23095 ms

5.	int main() { int a = 2;	35187 ms

	print();
	return 0;
	}	

6.	void print() {
	cout << "Hello World!" << endl;
	}
	
	int main() {
	int a = 2;
	print();
	return 0;
	}
		35871 ms

7.	#include<iostream>
	using namespace std;
	
	void print() {
	cout << "Hello World!" << endl;
	}
	
	int main() {
	int a = 2;
	print();
	return 0;
	}
		36117 ms

8.	#include<iostream>
	using namespace std;
	
	void print() {
	int k;
	string abc;
	double variable_random;
	cout << "Hello World!" << endl;
	}
	
	int main() {
	int a = 2;
	print();
	return 0;
	}
		43322 ms

Dari Tabel 5.38 kita bisa menyimpulkan bahwa dalam melakukan proses pemberian umpan balik pada penulisan kode program yang dilakukan oleh pengguna, sistem memerlukan waktu yang berbeda-beda tergantung dengan tingkat kesulitan *parsing* dari baris kode program menjadi AST. Selain itu, semakin banyak *identifier* yang akan dilakukan proses pengecekan *code styling convention*, maka semakin lama pula *running time* dari *Instant Feedback System* ini.

LAMPIRAN A – KODE SUMBER CONTROLLER

1. Kelas ConventionController yang menangani Manajemen Konvensi Gaya Penulisan Kode.

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Http\Requests;
use App\Convention;
use DB;

class ConventionController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $dbconventions = Convention::all();
        return view('convention.index', ['dbconventions' =>
            $dbconventions]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
```

```
    {
        $conventions = config('conventionmap');
        return view('convention.create', ['conventions' =>
            $conventions]);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $this->validate($request,[
            'for' => 'required',
            'regex' => 'required',
            'deskripsi' => 'required',
            'min' => 'required',
            'pesanmin' => 'required',
            ]);

        // input biasa
        $convention = new Convention;
        $convention->for = $request->for;
        $convention->regex = $request->regex;
        $convention->deskripsi = $request->deskripsi;
        $convention->min = $request->min;
        $convention->pesanmin = $request->pesanmin;
        $convention->save();

        return redirect('convention')->with('message', 'Konvensi
            kode baru berhasil ditambahkan');
    }
```

```
/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $convention = Convention::find($id);
    if(!$convention) {
        abort('404');
    }

    return view('convention.single')->with('convention',
        $convention);
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    $conventions = config('conventionmap');

    $convention = Convention::find($id);
    if(!$convention) {
        abort('404');
    }
```

```
    return view('convention.edit', ['conventions' =>
        $conventions, 'convention' => $convention]);
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    $this->validate($request,[
        'for' => 'required',
        'regex' => 'required',
        'deskripsi' => 'required',
        'min' => 'required',
        'pesanmin' => 'required',
        ]);

    // input biasa
    $convention = Convention::find($id);
    $convention->for = $request->for;
    $convention->regex = $request->regex;
    $convention->deskripsi = $request->deskripsi;
    $convention->min = $request->min;
    $convention->pesanmin = $request->pesanmin;
    $convention->save();
    return redirect('convention')->with('message', 'Konvensi
        kode berhasil diupdate');
}

/**
```

```
 * Remove the specified resource from storage.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
   $convention = Convention::find($id);
   try {
      $convention->delete();
   } catch (QueryException $e) {
      return redirect('convention')->with('error', 'Konvensi kode
        gagal dihapus, data masih direferensikan');
   }

   return redirect('convention')->with('message', 'Konvensi
        kode berhasil dihapus');
}

/**
 * Get the specified regex for code convention check.
 *
 * @param  string  $for
 * @return string  $regex
 */
public function getConventionRule($for)
{
   $regex = DB::table('elearning.convention')
   ->select('regex')
   ->where('for', '=', $for)
   ->get();

   return $regex;
}
```

```
    /**
     * Get the specified message for code convention check.
     *
     * @param  string  $for
     * @return string  $message
     */
    public function getConventionMessage($for)
    {
        $message = DB::table('elearning.convention')
        ->select('deskripsi')
        ->where('for', '=', $for)
        ->get();

        return $message;
    }

    /**
     * Get the specified minimal length for code convention check.
     *
     * @param  string  $for
     * @return string  $min
     */
    public function getConventionMinimal($for)
    {
        $min = DB::table('elearning.convention')
        ->select('min', 'pesanmin')
        ->where('for', '=', $for)
        ->get();

        return $min;
    }
}
```

Kode Sumber 7.1 Kelas ConventionController

2. Kelas CourseController yang menangani Manajemen Kursus.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use DB;

use App\Http\Requests;
use App\Course;
use App\Period;
use App\Subject;

use Illuminate\Database\QueryException;

class CourseController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $courses = DB::table('elearning.kursus as k')
                ->join('elearning.periode as p', 'p.id', '=',
            'k.periode_id')
                ->join('elearning.matakuliah as m', 'm.id', '=',
            'k.mk_id')
                ->select('k.*', 'p.nama as namaperiode', 'm.nama as
            namamatakuliah')
                ->get();
```

```
    return view('course.index', ['courses' => $courses]);
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    $periods = Period::all();
    $subjects = Subject::all();

    return view('course.create', ['periods' => $periods, 'subjects'
        => $subjects]);
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $this->validate($request,[
        'periode_id' => 'required',
        'mk_id' => 'required',
        'nama' => 'required',
        ]);

    // input biasa
    $course = new Course;
```

```
    $course->periode_id = $request->periode_id;
    $course->mk_id = $request->mk_id;
    $course->nama = $request->nama;
    $course->save();

    return redirect('course')->with('message', 'Kursus baru
        berhasil ditambahkan');
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $course = DB::table('elearning.kursus as k')
            ->join('elearning.periode as p', 'p.id', '=',
        'k.periode_id')
            ->join('elearning.matakuliah as m', 'm.id', '=',
        'k.mk_id')
            ->select('k.*', 'p.nama as namaperiode', 'm.nama as
        namamatakuliah')
            ->where('k.id', '=', $id)
            ->get();

    if(!$course) {
       abort('404');
    }

    return view('course.single')->with('course', $course);
}
```

```
    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $course = Course::find($id);
        $periods = Period::all();
        $subjects = Subject::all();

        if(!$course) {
            abort('404');
        }

        return view('course.edit', ['periods' => $periods, 'subjects' =>
              $subjects, 'course' => $course]);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        $this->validate($request,[
            'periode_id' => 'required',
            'mk_id' => 'required',
            'nama' => 'required',
            ]);
```

```
        // input biasa
        $course = Course::find($id);
        $course->periode_id = $request->periode_id;
        $course->mk_id = $request->mk_id;
        $course->nama = $request->nama;
        $course->save();

        return redirect('course')->with('message', 'Kursus berhasil
            diupdate');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $course = Course::find($id);

        try {
            $course->delete();
        } catch (QueryException $e) {
            return redirect('course')->with('error', 'Kursus gagal
              dihapus, data masih direferensikan');
        }

        return redirect('course')->with('message', 'Kursus berhasil
            dihapus');
    }
}
```

Kode Sumber 7.2 Kelas CourseController

3. Kelas EnrollController yang menangani Manajemen *Enrollment*.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Auth;
use App\Http\Requests;
use App\Enrollment;
use App\Course;
use DB;

use Illuminate\Database\QueryException;

class EnrollController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $enrolls = DB::table('elearning.enrollment')
        ->leftJoin('elearning.kursus', 'elearning.kursus.id', '=',
            'elearning.enrollment.kursus_id')
        ->select('elearning.kursus.*', 'elearning.enrollment.id as
            enrole_id')
        ->where('elearning.enrollment.user_id', '=', Auth::id())
        ->get();
```

```
    return view('enroll.index', ['enrolls' => $enrolls]);
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store($request)
{
    // input biasa
    $enroll = new Enrollment;
    $enroll->kursus_id = $request['kursus_id'];
    $enroll->user_id = $request['user_id'];
    $enroll->save();
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $course = Course::find($id);

    $enrolls = DB::table('users')
    ->leftJoin('elearning.enrollment', function($join) use($id) {
        $join->on('elearning.enrollment.user_id', '=', 'users.id');
        $join->on('elearning.enrollment.kursus_id', '=',
          DB::raw($id));
    })
```

```
        ->leftJoin('elearning.kursus', 'elearning.kursus.id', '=',
            'elearning.enrollment.kursus_id')
        ->leftJoin('role_user', 'role_user.user_id', '=', 'users.id')
        ->leftJoin('roles', 'roles.id', '=', 'role_user.role_id')
        ->select('users.name as namauser', 'users.email', 'roles.name
            as namarole', 'elearning.enrollment.id', 'users.id as
            userid')
        ->get();

        return view('enroll.single', ['enrolls' =>
            Enrollment::hydrate($enrolls), 'course' => $course]);
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $course = Course::find($id);

        $enrolls = DB::table('users')
        ->leftJoin('elearning.enrollment', function($join) use($id) {
            $join->on('elearning.enrollment.user_id', '=', 'users.id');
            $join->on('elearning.enrollment.kursus_id', '=',
            DB::raw($id));
        })
        ->leftJoin('elearning.kursus', 'elearning.kursus.id', '=',
            'elearning.enrollment.kursus_id')
        ->leftJoin('role_user', 'role_user.user_id', '=', 'users.id')
        ->leftJoin('roles', 'roles.id', '=', 'role_user.role_id')
```

```
    ->select('users.name as namauser', 'users.email', 'roles.name
        as namarole', 'elearning.enrollment.id', 'users.id as
        userid')
    ->get();

    return view('enroll.edit', ['enrolls' =>
        Enrollment::hydrate($enrolls), 'course' => $course]);
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    foreach ($request['user_id'] as $user_id) {
        $isexist = DB::table('elearning.enrollment')
        ->select('enrollment.id')
        ->where('user_id', '=', $user_id)
        ->where('kursus_id', '=', $id)
        ->first();

        $record = array();
        $record['kursus_id'] = $id;
        $record['user_id'] = $user_id;

        if(!empty($request['data::'.$user_id]) &&
          empty($isexist)) {
            self::store($record);
        }
```

```
            else if(empty($request['data::'.$user_id])
              && !empty($isexist)) {
                $err = self::destroy($isexist->id);
                if($err) {
                    return redirect('enroll/'.$id)->with('error', 'Enrollment
              gagal dihapus, data masih direferensikan');
                }
            }
        }

        return redirect('enroll/'.$id)->with('message', 'Enrollment
            berhasil diupdate');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $enroll = Enrollment::find($id);

        try {
            $enroll->delete();
        } catch (QueryException $e) {
            return true;
        }

        return false;
    }
}
```

Kode Sumber 7.3 Kelas EnrollController

4. Kelas QuizController yang menangani Manajemen Penugasan.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Requests;
use App\User;
use App\Enrollment;
use App\Course;
use App\Quiz;
use Auth;
use DB;
use Illuminate\Database\QueryException;

class QuizController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index($id)
    {
        $enroll = Enrollment::find($id);
        $course = Course::find($enroll->kursus_id);
        $quizes = DB::table('elearning.tugas')
            ->leftJoin('elearning.enrollment', 'elearning.enrollment.id',
              '=', 'elearning.tugas.enroll_id')
            ->leftJoin('elearning.kursus', 'elearning.kursus.id', '=',
              'elearning.enrollment.kursus_id')
            ->select('elearning.tugas.*')
```

```
        ->where('elearning.kursus.id', '=', $course->id)
        ->get();

    $ismhs = false;
    $user = User::find(Auth::id());
    if($user->is('mhs')) {
        $ismhs = true;
    }
    return view('quiz.index', ['course' => $course, 'quizes' =>
        $quizes, 'enrollid' => $id, 'ismhs' => $ismhs]);
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create($id)
{
    return view('quiz.create', ['enrollid' => $id]);
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request, $id)
{
    $this->validate($request, [
        'nama' => 'required',
        'waktupengerjaan' => 'required',
        'des' => 'required',
```

```
        'jwb' => 'required',
    ]);

    // input biasa
    $quiz = new Quiz;
    $quiz->enroll_id = $id;
    $quiz->nama = $request->nama;
    $quiz->wmulai = explode(' - ', $request-
        >waktupengerjaan)[0];
    $quiz->wselesai = explode(' - ', $request-
        >waktupengerjaan)[1];
    $quiz->des = $request->des;
    $quiz->jwb = $request->jwb;
    $quiz->save();
    return redirect('enroll/'.$id.'/quiz/'.$quiz_id)-
        >with('message', 'Penugasan baru berhasil
        ditambahkan');
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id, $quiz_id)
{
    $quiz = Quiz::find($quiz_id);

    $ismhs = false;
    $user = User::find(Auth::id());
    if($user->is('mhs')) {
        $ismhs = true;
    }
```

```
    return view('quiz.single', ['quiz' => $quiz, 'enrollid' => $id,
        'ismhs' => $ismhs]);
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function edit($id, $quiz_id)
{
    $quiz = Quiz::find($quiz_id);
    return view('quiz.edit', ['enrollid' => $id, 'quizid' =>
        $quiz_id, 'quiz' => $quiz]);
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id, $quiz_id)
{
$this->validate($request, [
        'nama' => 'required',
        'waktupengerjaan' => 'required',
        'des' => 'required',
        'jwb' => 'required',
    ]);
```

```
        // input biasa
        $quiz = Quiz::find($quiz_id);
        $quiz->nama = $request->nama;
        $quiz->wmulai = explode(' - ', $request-
            >waktupengerjaan)[0];
        $quiz->wselesai = explode(' - ', $request-
            >waktupengerjaan)[1];
        $quiz->des = $request->des;
        $quiz->jwb = $request->jwb;
        $quiz->save();
        return redirect('enroll/'.$id.'/quiz/'.$quiz_id)-
            >with('message', 'Penugasan berhasil diupdate');
    }
    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id, $quiz_id)
    {
        $quiz = Quiz::find($quiz_id);

        try {
            $quiz->delete();
        } catch (QueryException $e) {
            return redirect('enroll/'.$id.'/quiz')->with('error',
             'Penugasan gagal dihapus, data masih direferensikan');
        }
        return redirect('enroll/'.$id.'/quiz')->with('message',
            'Penugasan berhasil dihapus');
    }
}
```

Kode Sumber 7.4 Kelas QuizController

5. Kelas AnswerController yang menangani Manajemen Jawaban.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;
use App\User;
use App\Enrollment;
use App\Course;
use App\Quiz;
use App\Answer;
use App\Detail;
use Auth;
use DB;

class AnswerController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index($id, $quiz_id)
    {
        $quiz = Quiz::find($quiz_id);

        $ismhs = false;
        $user = User::find(Auth::id());
        if($user->is('mhs')) {
            $ismhs = true;
```

```
    }

    $enroll = Enrollment::find($id);
    $course = Course::find($enroll->kursus_id);

    if($ismhs == true) {
       $participants = DB::table('elearning.enrollment')
       ->leftJoin('users', 'users.id', '=',
        'elearning.enrollment.user_id')
       ->leftJoin('elearning.pengumpulan', function($join)
        use($quiz_id) {
          $join->on('elearning.pengumpulan.enroll_id', '=',
        'elearning.enrollment.id');
          $join->on('elearning.pengumpulan.tugas_id', '=',
        DB::raw($quiz_id));
       })
       ->select('users.name as username',
        'elearning.pengumpulan.id as answerid')
       ->where('elearning.enrollment.kursus_id', '=', $course-
        >id)
       ->where('elearning.enrollment.user_id', '=', Auth::id())
       ->get();

    }
    else {
       $participants = DB::table('elearning.enrollment')
       ->leftJoin('users', 'users.id', '=',
        'elearning.enrollment.user_id')
       ->leftJoin('elearning.pengumpulan', function($join)
        use($quiz_id) {
          $join->on('elearning.pengumpulan.enroll_id', '=',
        'elearning.enrollment.id');
          $join->on('elearning.pengumpulan.tugas_id', '=',
        DB::raw($quiz_id));
```

```
        })
        ->select('users.name as username',
          'elearning.pengumpulan.id as answerid')
        ->where('elearning.enrollment.kursus_id', '=', $course-
          >id)
        ->where('elearning.enrollment.user_id', '!=', Auth::id())
        ->get();

    }

    return view('answer.index', ['participants' =>
        User::hydrate($participants), 'course' => $course, 'quiz'
        => $quiz, 'enrollid' => $id, 'quizid' => $quiz_id, 'ismhs'
        => $ismhs]);
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create($id, $quiz_id)
{
    $quiz = Quiz::find($quiz_id);
    return view('answer.create', ['enrollid' => $id, 'quizid' =>
        $quiz_id, 'quiz' => $quiz]);
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
```

```
public function store(Request $request, $id, $quiz_id)
{
    // $this->validate($request, [
    //     'kode' => 'required',
    //     'durasi' => 'required',
    //     'errsyntax' => 'required',
    //     'errconvention' => 'required',
    // ]);
    $answer = new Answer;
    $answer->enroll_id = $id;
    $answer->tugas_id = $quiz_id;
    $answer->save();

    // input biasa
    $detail = new Detail;
    $detail->kumpul_id = $answer->id;
    $time = explode(":", $request->durasi);
    $durasi = $time[0] * 60;
    $durasi = $durasi + ($time[1] * 1);
    $detail->durasi = $durasi;
    $detail->errsyntax = $request->errsyntax;
    $detail->errconvention = $request->errconvention;
    $detail->kode = $request->kode;
    $detail->save();

    return
        redirect('enroll/'.$id.'/quiz/'.$quiz_id.'/answer/'.$answer-
        >id)->with('message', 'Jawaban baru berhasil
        ditambahkan');
}

/**
 * Display the specified resource.
 *
```

```
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id, $quiz_id, $answer_id)
    {
        $enroll = Enrollment::find($id);
        $course = Course::find($enroll->kursus_id);
        $user = User::find($enroll->user_id);
        $detail = DB::table('elearning.detail')
        ->select('elearning.detail.*')
        ->where('kumpul_id', '=', $answer_id)
        ->get();

        $detailpercobaan = array();
        $detaildurasi = array();
        $detailconventionerr = array();
        $detailsyntaxerr = array();
        foreach ($detail as $key => $value) {
            $detailpercobaan[] = $key;
            $detaildurasi[] = $value->durasi;
            $detailconventionerr[] = $value->errconvention;
            $detailsyntaxerr[] = $value->errsyntax;
        }

        $quiz = Quiz::find($quiz_id);
        $answer = Answer::find($answer_id);
        return view('answer.single', ['enrollid' => $id, 'quizid' =>
            $quiz_id, 'answerid' => $answer_id, 'course' => $course,
            'quiz' => $quiz, 'answer' => $answer, 'user' => $user,
            'detailpercobaan' => json_encode($detailpercobaan),
            'detaildurasi' => json_encode($detaildurasi),
            'detailconventionerr' =>
            json_encode($detailconventionerr), 'detailsyntaxerr' =>
            json_encode($detailsyntaxerr)]);
```

```
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id, $quiz_id, $answer_id)
    {
        $quiz = Quiz::find($quiz_id);
        $answer = Answer::find($answer_id);
        return view('answer.edit', ['enrollid' => $id, 'quizid' =>
            $quiz_id, 'answerid' => $answer_id, 'quiz' => $quiz,
            'answer' => $answer]);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id, $quiz_id,
            $answer_id)
    {
        // $this->validate($request, [
        //     'kode' => 'required',
        //     'durasi' => 'required',
        //     'errsyntax' => 'required',
        //     'errconvention' => 'required',
        // ]);
```

```
        // input biasa
        $detail = new Detail;
        $detail->kumpul_id = $answer_id;
        $time = explode(":", $request->durasi);
        $durasi = $time[0] * 60;
        $durasi = $durasi + ($time[1] * 1);
        $detail->durasi = $durasi;
        $detail->errsyntax = $request->errsyntax;
        $detail->errconvention = $request->errconvention;
        $detail->kode = $request->kode;
        $detail->save();

        return
            redirect('enroll/'.$id.'/quiz/'.$quiz_id.'/answer/'.$answer_
            id)->with('message', 'Jawaban berhasil diupdate');
    }
}
```

Kode Sumber 7.5 Kelas AnswerController

LAMPIRAN B – METHOD-METHOD LISTENER

Method-method yang ditulis di bawah ini adalah *method-method* turunan yang berguna untuk menelusuri *node* di dalam AST.

```
StatementPrinter = function(annotations) {
   CPP14Listener.CPP14Listener.call(this);
   this.annotations = annotations;
   return this;
};

StatementPrinter.prototype =
         Object.create(CPP14Listener.CPP14Listener.prototype);
StatementPrinter.prototype.constructor = StatementPrinter;

var typedefpotential = false;
var constpotensial = false;
StatementPrinter.prototype.enterDeclspecifier = function (ctx) {
   console.log('enterDeclspecifier');
   console.log(ctx.getText());
   if(ctx.getText() == 'typedef') {
      typedefpotential = true;
   } else if(ctx.getText() == 'const') {
      constpotensial = true;
   }
}

// wrong output
StatementPrinter.prototype.enterConstantexpression = function
         (ctx) {
   console.log('enterConstantexpression');
```

```
    console.log(ctx.getText());
}

StatementPrinter.prototype.enterNamespacedefinition = function
        (ctx) {
    var sfor = 'namespace';
    var text = ctx.getText();
    var deletednamespace = text.replace('namespace', '');
    var namespacename = deletednamespace.split('{');
    ConventionCheck(sfor, ctx, this.annotations,
        namespacename[0]);
}

StatementPrinter.prototype.enterNamespacename = function (ctx)
        {
    var sfor = 'namespace';
    ConventionCheck(sfor, ctx, this.annotations);
}

var variablepotential = false;
var initialchar = '';
StatementPrinter.prototype.enterSimpledeclaration = function
        (ctx) {
    var text = ctx.getText();
    console.log('enterSimpledeclaration');
    console.log(ctx.getText());
    variablepotential = true;
    initialchar = text.substr(0, 1);
}

StatementPrinter.prototype.exitSimpledeclaration = function (ctx)
        {
    console.log('exitSimpledeclaration');
    console.log(ctx.getText());
```

```
    variablepotential = false;
    typedefpotential = false;
    initialchar = '';
}

StatementPrinter.prototype.enterMemberdeclarator = function
            (ctx) {
    variablepotential = false;
    console.log('enterMemberdeclarator');
    console.log(ctx.getText());
    console.log(initialchar);
    if(initialchar == 'c') {
        var sfor = 'classdatamember';
    } else if(initialchar == 's') {
        var sfor = 'structdatamember';
    }
    ConventionCheck(sfor, ctx, this.annotations);
}

StatementPrinter.prototype.enterEnumhead = function (ctx) {
    var sfor = 'enum';
    var text = ctx.getText();
    var enumname = text.replace('enum', '');
    ConventionCheck(sfor, ctx, this.annotations, enumname);
}

StatementPrinter.prototype.enterEnumerator = function (ctx) {
    var sfor = 'enumerator';
    ConventionCheck(sfor, ctx, this.annotations);
}

StatementPrinter.prototype.enterAliasdeclaration = function (ctx)
            {
    var sfor = 'alias';
```

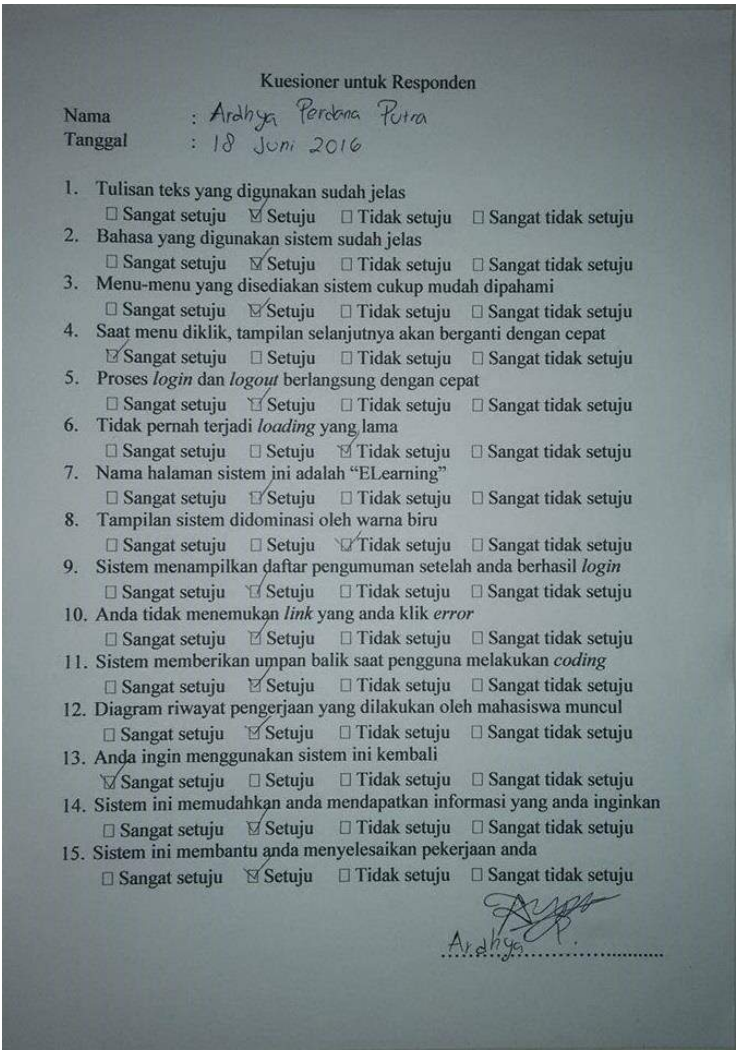
```
    var text = ctx.getText();
    var usingdeleted = text.replace('using', '');
    var aliasname = usingdeleted.split('=');
    ConventionCheck(sfor, ctx, this.annotations, aliasname[0]);
}

StatementPrinter.prototype.enterClassname = function (ctx) {
    var sfor = 'classandstruct';
    ConventionCheck(sfor, ctx, this.annotations);
};

StatementPrinter.prototype.enterDeclarator = function (ctx) {
    console.log('enterDeclarator');
    console.log(ctx.getText());
    var text = ctx.getText();
    if(text.indexOf('main(') < 0) {
        if(text.indexOf(')') > -1) {
            var splitedfunction = text.split("(");
            var functionname = splitedfunction[0];
            var sfor = 'function';
            ConventionCheck(sfor,          ctx,          this.annotations,
              functionname);
        } else {
            if(typedefpotential == false && constpotensial == false) {
                if(variablepotential == true) {
                    var sfor = 'commonvariable';
                    ConventionCheck(sfor, ctx, this.annotations);
                }
            } else if(typedefpotential == true) {
                var sfor = 'typedef';
                ConventionCheck(sfor, ctx, this.annotations);
            } else if(constpotensial == true) {
                var sfor = 'constant';
                ConventionCheck(sfor, ctx, this.annotations);
```

```
        }
    }
  }
};

StatementPrinter.prototype.exitDeclarator = function (ctx) {
  var text = ctx.getText();
  if(text.indexOf('main(') < 0) {
    if(text.indexOf(')') > -1) {
      islocal = false;
    }
  }
};
```

Kode Sumber 8.1 Beberapa *Method Listener* yang Digunakan Pada Sistem

[Halaman ini sengaja dikosongkan]

LAMPIRAN C – KUESIONER *USABILITY TEST*



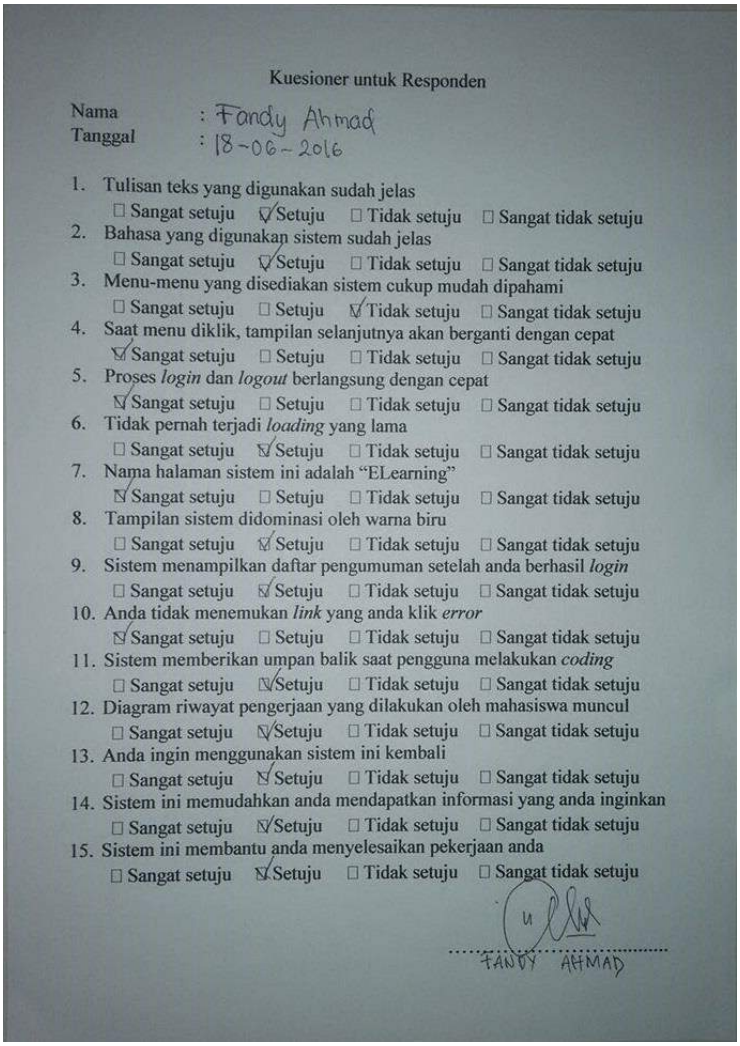
Kuesioner untuk Responden

Nama : Ardhya Perdana Putra
Tanggal : 18 Juni 2016

1. Tulisan teks yang digunakan sudah jelas
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
2. Bahasa yang digunakan sistem sudah jelas
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
3. Menu-menu yang disediakan sistem cukup mudah dipahami
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
4. Saat menu diklik, tampilan selanjutnya akan berganti dengan cepat
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
5. Proses *login* dan *logout* berlangsung dengan cepat
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
6. Tidak pernah terjadi *loading* yang lama
   ☐ Sangat setuju ☐ Setuju ☑ Tidak setuju ☐ Sangat tidak setuju
7. Nama halaman sistem ini adalah "ELearning"
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
8. Tampilan sistem didominasi oleh warna biru
   ☐ Sangat setuju ☐ Setuju ☑ Tidak setuju ☐ Sangat tidak setuju
9. Sistem menampilkan daftar pengumuman setelah anda berhasil *login*
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
10. Anda tidak menemukan *link* yang anda klik *error*
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
11. Sistem memberikan umpan balik saat pengguna melakukan *coding*
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
12. Diagram riwayat pengerjaan yang dilakukan oleh mahasiswa muncul
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
13. Anda ingin menggunakan sistem ini kembali
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
14. Sistem ini memudahkan anda mendapatkan informasi yang anda inginkan
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
15. Sistem ini membantu anda menyelesaikan pekerjaan anda
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju

Ardhya

Gambar 9.1 Respoden 1 Ardhya Perdana Putra



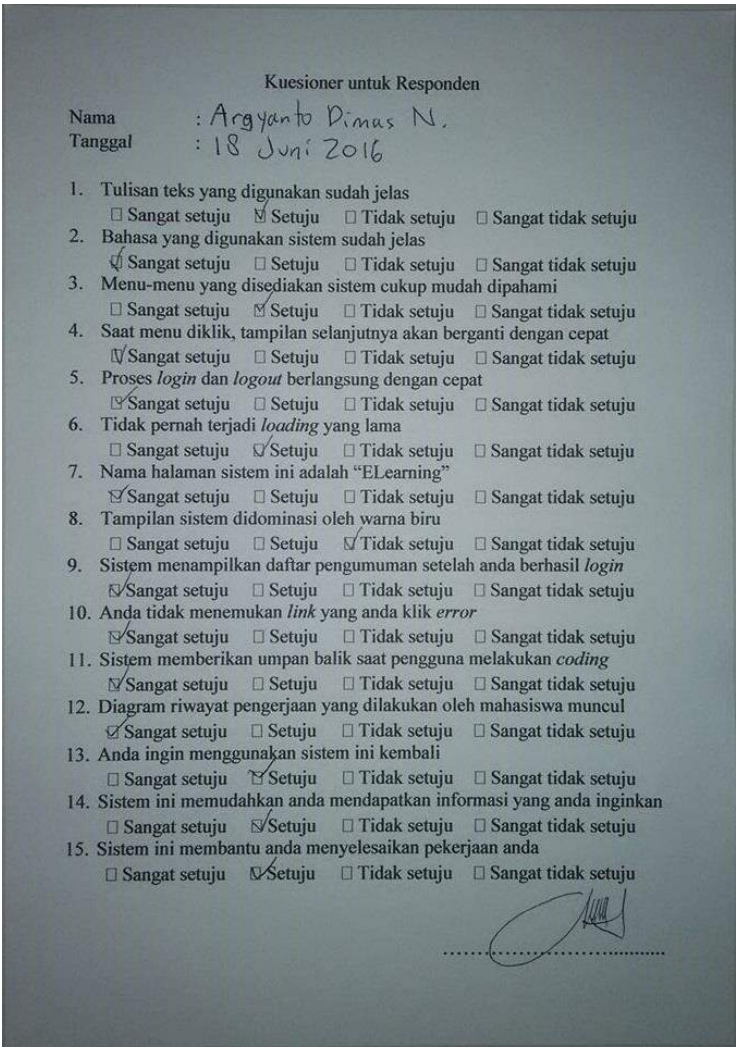
Kuesioner untuk Responden

Nama : Fandy Ahmad
Tanggal : 18-06-2016

1. Tulisan teks yang digunakan sudah jelas
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
2. Bahasa yang digunakan sistem sudah jelas
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
3. Menu-menu yang disediakan sistem cukup mudah dipahami
   ☐ Sangat setuju ☐ Setuju ☑ Tidak setuju ☐ Sangat tidak setuju
4. Saat menu diklik, tampilan selanjutnya akan berganti dengan cepat
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
5. Proses *login* dan *logout* berlangsung dengan cepat
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
6. Tidak pernah terjadi *loading* yang lama
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
7. Nama halaman sistem ini adalah "ELearning"
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
8. Tampilan sistem didominasi oleh warna biru
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
9. Sistem menampilkan daftar pengumuman setelah anda berhasil *login*
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
10. Anda tidak menemukan *link* yang anda klik *error*
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
11. Sistem memberikan umpan balik saat pengguna melakukan *coding*
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
12. Diagram riwayat pengerjaan yang dilakukan oleh mahasiswa muncul
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
13. Anda ingin menggunakan sistem ini kembali
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
14. Sistem ini memudahkan anda mendapatkan informasi yang anda inginkan
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
15. Sistem ini membantu anda menyelesaikan pekerjaan anda
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju

FANDY AHMAD

Gambar 9.2 Responden 2 Fandy Ahmad



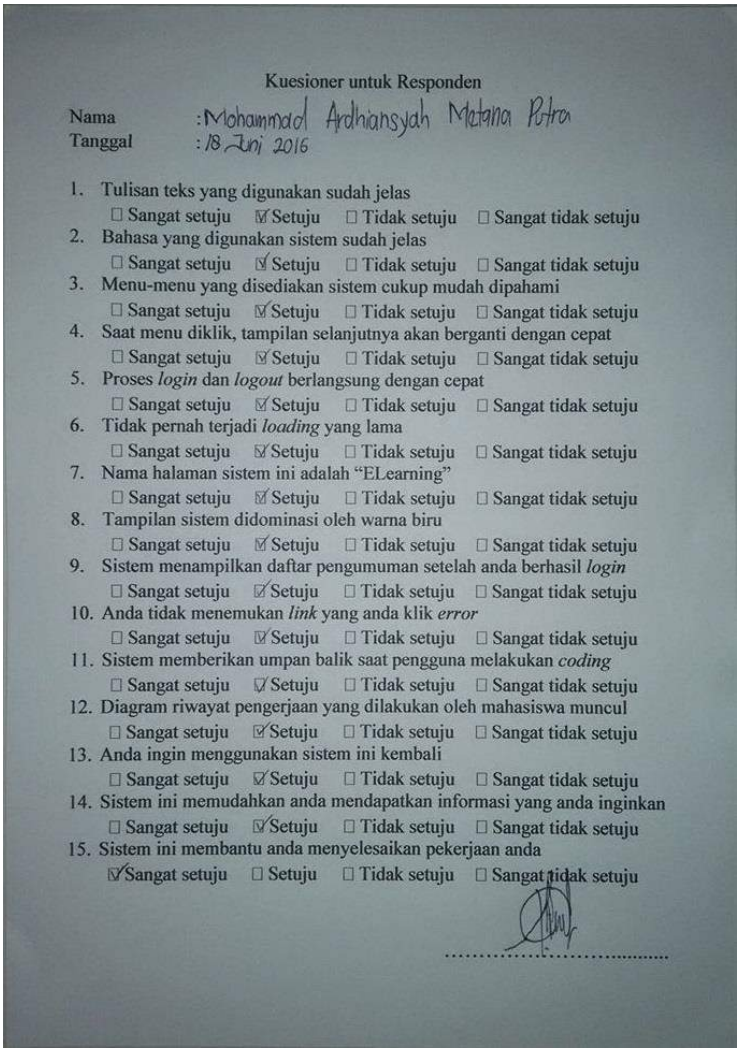
Kuesioner untuk Responden

Nama : Argyanto Dimas N.
Tanggal : 18 Juni 2016

1. Tulisan teks yang digunakan sudah jelas
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
2. Bahasa yang digunakan sistem sudah jelas
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
3. Menu-menu yang disediakan sistem cukup mudah dipahami
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
4. Saat menu diklik, tampilan selanjutnya akan berganti dengan cepat
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
5. Proses *login* dan *logout* berlangsung dengan cepat
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
6. Tidak pernah terjadi *loading* yang lama
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
7. Nama halaman sistem ini adalah "ELearning"
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
8. Tampilan sistem didominasi oleh warna biru
   ☐ Sangat setuju ☐ Setuju ☑ Tidak setuju ☐ Sangat tidak setuju
9. Sistem menampilkan daftar pengumuman setelah anda berhasil *login*
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
10. Anda tidak menemukan *link* yang anda klik *error*
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
11. Sistem memberikan umpan balik saat pengguna melakukan *coding*
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
12. Diagram riwayat pengerjaan yang dilakukan oleh mahasiswa muncul
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
13. Anda ingin menggunakan sistem ini kembali
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
14. Sistem ini memudahkan anda mendapatkan informasi yang anda inginkan
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
15. Sistem ini membantu anda menyelesaikan pekerjaan anda
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju

.....

Gambar 9.3 Responden 3 Argyanto Dimas N.



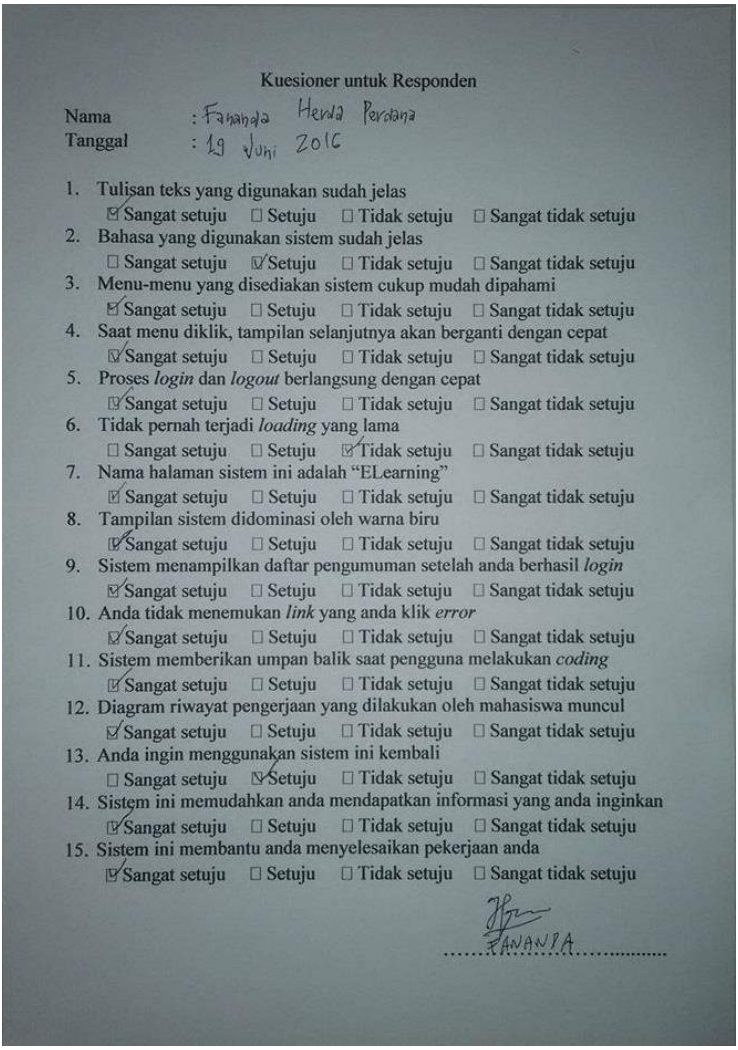
Kuesioner untuk Responden

Nama : Mohammad Ardhiansyah Metana Putra
Tanggal : 18 Juni 2016

1. Tulisan teks yang digunakan sudah jelas
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
2. Bahasa yang digunakan sistem sudah jelas
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
3. Menu-menu yang disediakan sistem cukup mudah dipahami
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
4. Saat menu diklik, tampilan selanjutnya akan berganti dengan cepat
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
5. Proses *login* dan *logout* berlangsung dengan cepat
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
6. Tidak pernah terjadi *loading* yang lama
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
7. Nama halaman sistem ini adalah "ELearning"
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
8. Tampilan sistem didominasi oleh warna biru
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
9. Sistem menampilkan daftar pengumuman setelah anda berhasil *login*
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
10. Anda tidak menemukan *link* yang anda klik *error*
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
11. Sistem memberikan umpan balik saat pengguna melakukan *coding*
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
12. Diagram riwayat pengerjaan yang dilakukan oleh mahasiswa muncul
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
13. Anda ingin menggunakan sistem ini kembali
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
14. Sistem ini memudahkan anda mendapatkan informasi yang anda inginkan
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
15. Sistem ini membantu anda menyelesaikan pekerjaan anda
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju

.....

Gambar 9.4 Responden 4 M. Ardhiansyah Metana P.



Kuesioner untuk Responden

Nama : Fananda Herda Perdana
Tanggal : 19 Juni 2016

1. Tulisan teks yang digunakan sudah jelas
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
2. Bahasa yang digunakan sistem sudah jelas
   ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
3. Menu-menu yang disediakan sistem cukup mudah dipahami
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
4. Saat menu diklik, tampilan selanjutnya akan berganti dengan cepat
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
5. Proses *login* dan *logout* berlangsung dengan cepat
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
6. Tidak pernah terjadi *loading* yang lama
   ☐ Sangat setuju ☐ Setuju ☑ Tidak setuju ☐ Sangat tidak setuju
7. Nama halaman sistem ini adalah "ELearning"
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
8. Tampilan sistem didominasi oleh warna biru
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
9. Sistem menampilkan daftar pengumuman setelah anda berhasil *login*
   ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
10. Anda tidak menemukan *link* yang anda klik *error*
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
11. Sistem memberikan umpan balik saat pengguna melakukan *coding*
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
12. Diagram riwayat pengerjaan yang dilakukan oleh mahasiswa muncul
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
13. Anda ingin menggunakan sistem ini kembali
    ☐ Sangat setuju ☑ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
14. Sistem ini memudahkan anda mendapatkan informasi yang anda inginkan
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju
15. Sistem ini membantu anda menyelesaikan pekerjaan anda
    ☑ Sangat setuju ☐ Setuju ☐ Tidak setuju ☐ Sangat tidak setuju

FANANDA

Gambar 9.5 Responden 5 Fananda Herda Perdana

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1. Kesimpulan

Dari aplikasi dari tugas akhir ini, maka dapat disimpulkan bahwa:

a. *Elearning* dengan model *platform modular* sudah dibuat dan berfungsi dengan baik. Modul eksternal seperti *Student Feedback System* dan *Plagiarism Detection System* bisa diintegrasikan dengan *elearning*. Selain itu *elearning* juga memiliki kegunaan dalam aspek *learnability*, *efficiency*, *memorability*, *error*, dan *satisfaction* yang baik.
b. Modul manajemen kelas yang ada di dalam *elearning* sudah dibangun dan berjalan dengan baik. Modul ini bisa mengatur aktifitas *enroll* dari dosen dan mahasiswa dan juga bisa mengatur alur penugasan dan mengumpulan jawaban.
c. Modul umpan balik instan sudah dibangun. Namun kecepatan pemberian umpan balik tergantung pada tinggi atau rendahnya tingkat kompleksitas unsur-unsur dari penyusun kode program. Hal ini menyebabkan pemberian umpan balik terkadang tidak secepat yang diinginkan. Selain itu, penggunaan ANTLR JavaScript Target menyebabkan *browser* memakan sumber daya memori lebih banyak dan apabila terlalu parah bisa menyebabkan *force close* pada *browser*.

6.2. Saran

Saran yang dapat digunakan untuk penelitian lebih lanjut adalah sebagai berikut:

a. Sistem bisa dikembangkan menjadi lebih fleksibel lagi dalam mendeteksi bahasa pemrograman yang lainnya.
b. *Platform elearning* bisa dikembangkan lagi ke versi yang lebih stabil, seperti penambahan fitur notifikasi, *messaging*, dan fitur-fitur pendukung lainnya.

DAFTAR PUSTAKA

[1] Wikipedia, “Abstract syntax tree,” Wikimedia Foundation, Inc., 24 November 2015. [Online]. Available: https://en.wikipedia.org/wiki/Abstract_syntax_tree. [Diakses 13 December 2015].

[2] Cloud9 IDE and Mozilla, “Ace,” Cloud9 IDE and Mozilla, [Online]. Available: https://ace.c9.io. [Diakses 13 December 2015].

[3] ANTLR, “ANTLR,” ANTLR, 2014. [Online]. Available: http://www.antlr.org. [Diakses 13 December 2015].

[4] T. Otwell, “Laravel,” 2015. [Online]. Available: http://laravel.com/. [Diakses 16 December 2015].

[5] Almsaeed Studio, “Free Bootstrap Admin Template,” Almsaeed Studio, 2016. [Online]. Available: https://almsaeedstudio.com/. [Diakses 1 April 2016].

[6] W3C, “Web Worker,” W3C, 24 September 2015. [Online]. Available: https://www.w3.org/TR/workers/. [Diakses 1 April 2016].

[7] W3Schools, “HTML5 Web Workers,” W3Schools, [Online]. Available: http://www.w3schools.com/html/html5_webworkers.asp. [Diakses 1 April 2016].

[8] J. Goyvaerts, “Regular-Expressions.info,” 6 January 2016. [Online]. Available: http://www.regular-expressions.info/. [Diakses 1 April 2016].

[9] The PostgreSQL Global Development Group, “PostgreSQL,” The PostgreSQL Global Development Group, 2016. [Online]. Available: https://www.postgresql.org/. [Diakses 1 April 2016].

[10] R. Ilavi, Rancang Bangun E-Learning Pemrograman pada Modul Student Feedback System, Surabaya, 2016.

[11] R. I. Tantra, Rancang Bangun Sistem E-learning Pemrograman pada Modul Deteksi Plagiarisme Kode Program Mahasiswa dalam Satu Kelas, Surabaya, 2016.

[12] T. Parr, The Definitive ANTLR 4 Reference, November: The Pragmatic Programmers, LLC., 2016.

[13] H. Azmi, “Usulan Tugas Akhir,” Surabaya, 2016.

[14] M. Z. Sauqi, Kakas Pemantauan Kinerja Programmer untuk Peningkatan Proses Personal, Surabaya, 2015.

BIODATA PENULIS

Hafidh Azmi, biasa dipanggil Hafidh, dilahirkan di kota Trenggalek pada tanggal 7 September 1994. Penulis adalah anak pertama dari dua bersaudara dan dibesarkan di kota Trenggalek, Jawa Timur. Penulis menempuh pendidikan di SDN 1 Malasan, Trenggalek, SMP Negeri 1 Kauman, Tulungagung dan SMA Negeri 1 Boyolangu, Tulungagung. Pada tahun 2012, penulis mengikuti SNMPTN Tulis dan diterima di Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya yang terdaftar dengan NRP 5112100096. Di jurusan Teknik Informatika ini, penulis mengambil rumpun mata kuliah Algoritma dan Pemrograman. Selama di kuliah, penulis banyak belajar mengenai pemrograman berbasis web, pemrograman perangkat bergerak, dan pernah menciptakan beberapa permainan. Penulis juga pernah menjadi assiten mata kuliah Sistem Basis Data.

Selain di bidang akademik, penulis juga aktif di beberapa organisasi di antaranya Staf Kewirausahaan dan Minat Bakat HMTC Bersahabat, Staf Hubungan Masyarakat KMI Keluarga Tauhid, Staf Dana dan Usaha Schematics 2013, Staf Ahli Kewirausahaan dan Minat Bakat HMTC Berkarya, Staf Ahli Hubungan Kelembagaan KMI Santun Bersahaja, dan Staf Dana Schematics 2014. Saat buku ini ditulis, penulis sedang menimba ilmu di Sentra Vidya Utama sebagai seorang programmer paruh waktu yang ditempatkan di Tim Pengembangan Sistem Informasi Kepegawaian. Penulis juga bisa dihubungi melalui email hafidhazmi026@gmail.com